Cloud Aware Application Development Ecosystem

Jun 07, 2019

Introduction

CHAPTER 1

Architectural Overview

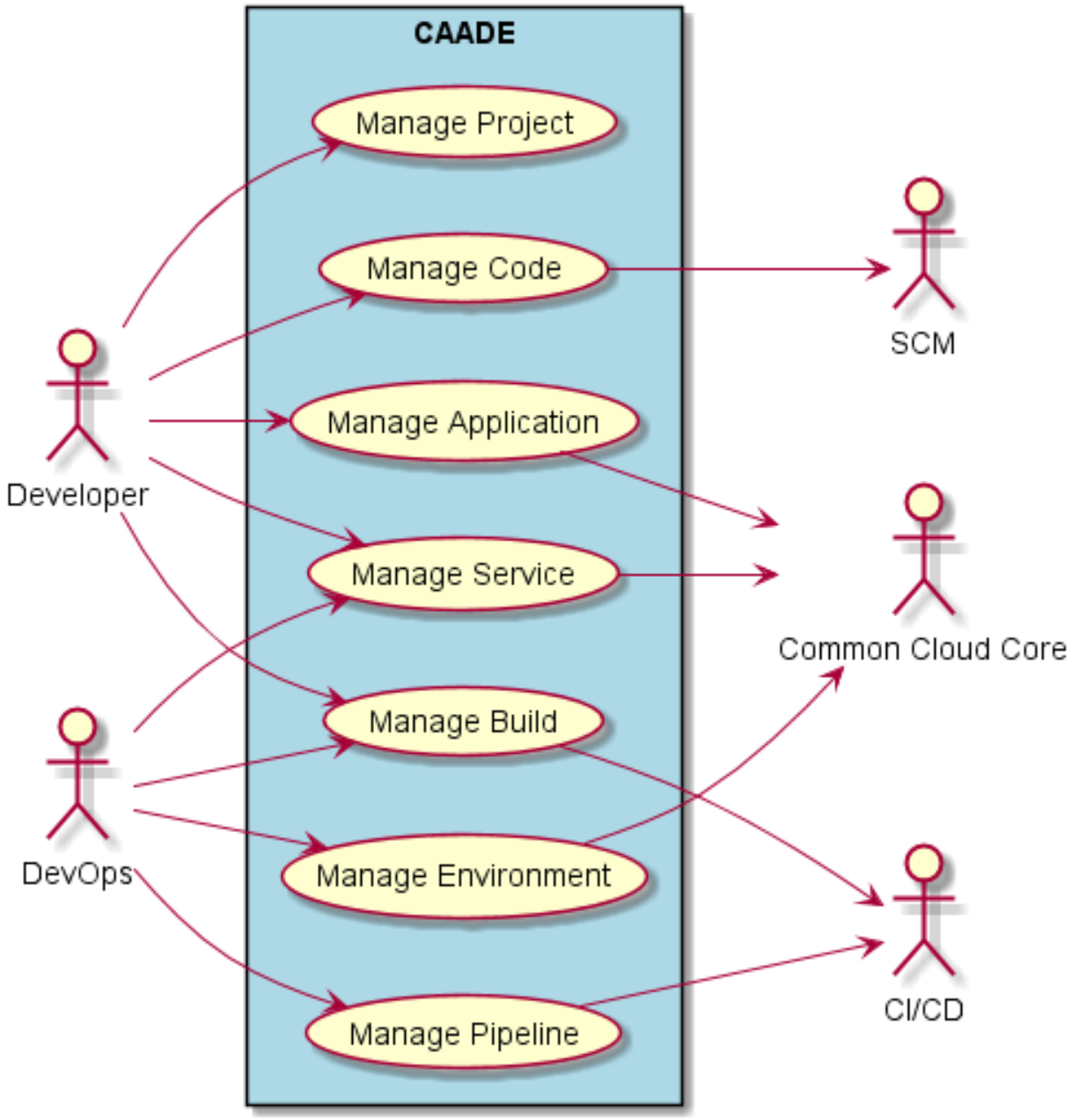
CAADE (Cloud Aware Application Development Environment) is a reference architecture that discribes how developers and devops work together in a hybrid cloud to enable the development of cloud aware applications.

1.1 Users

- *Developer*
- *DevOps*

1.2 High level Use Cases

- *Manage Application*
- *Manage Build*
- *Manage Code*
- *Manage Environment*
- *Manage Pipeline*
- *Manage Project*
- *Manage Service*

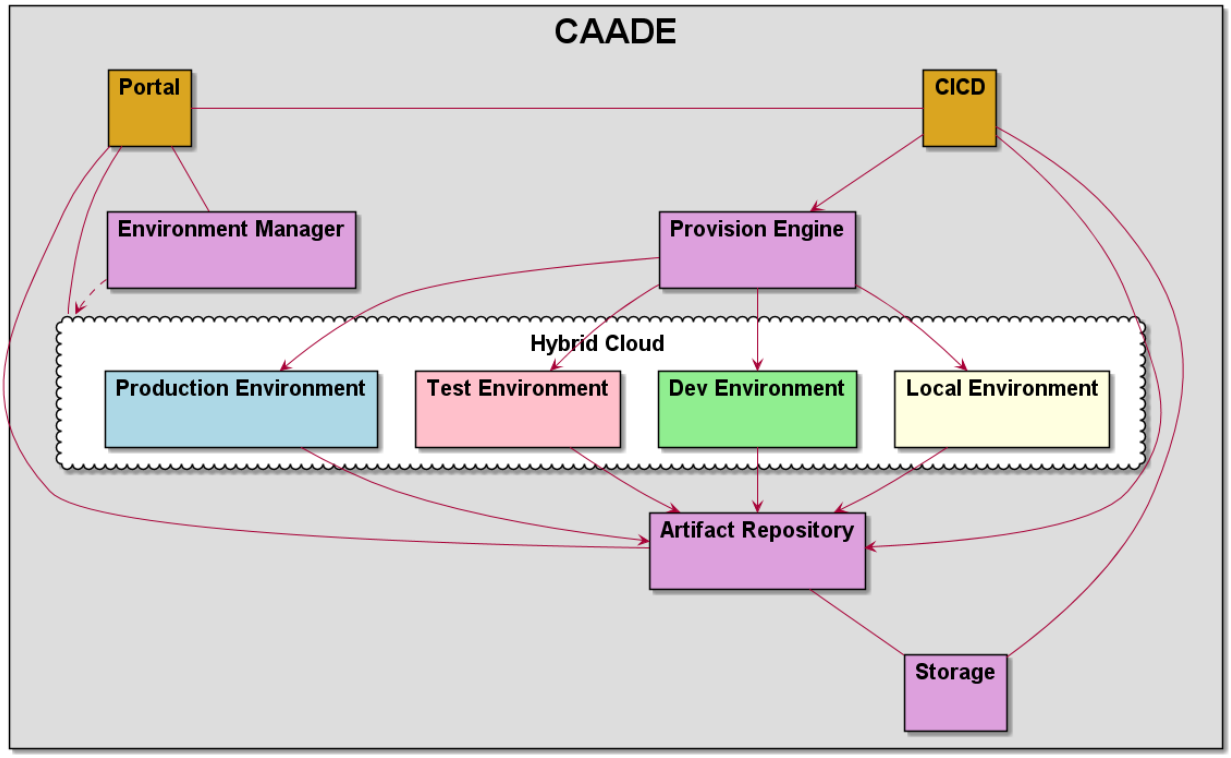


1.3 Logical Architecture

Developers need to focus on the developement of applications. When code is modified and checked into a code repository like github. A CI/CD system will automatically build, test and deploy the application, microservice or project. Multiple environments that have been created in the Common Cloud Core will be used by CAADE and the CI/CD to promote applications across the different environments.

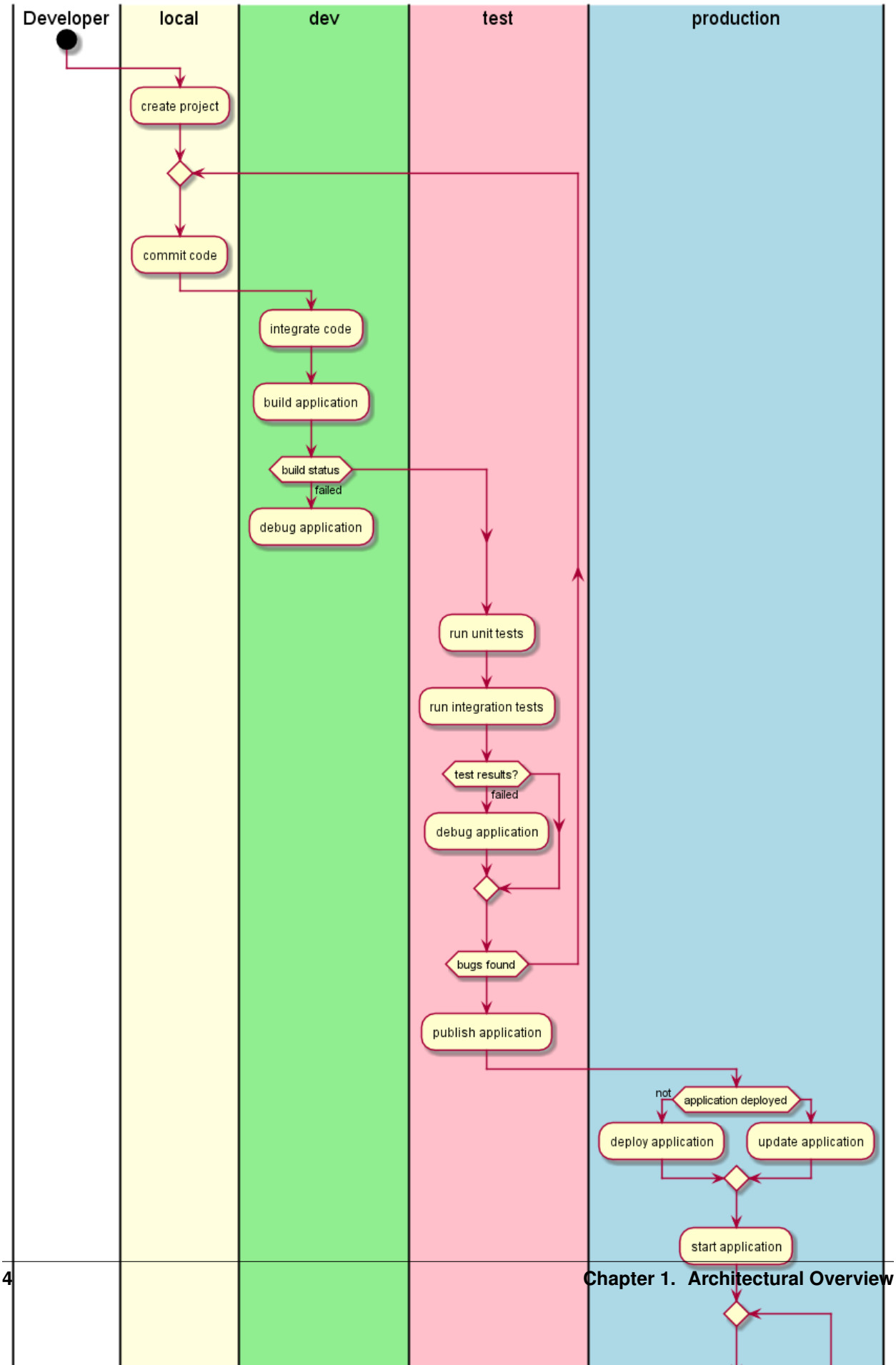
- *Automation Framework*
- *CICD*

- *Hybrid Cloud*
- *Hybrid Cloud / Artifact Repository*
- *Hybrid Cloud/Environment Manager*
- *Hybrid Cloud/Provision Engine*
- *Hybrid Cloud/Storage*
- *SCM*



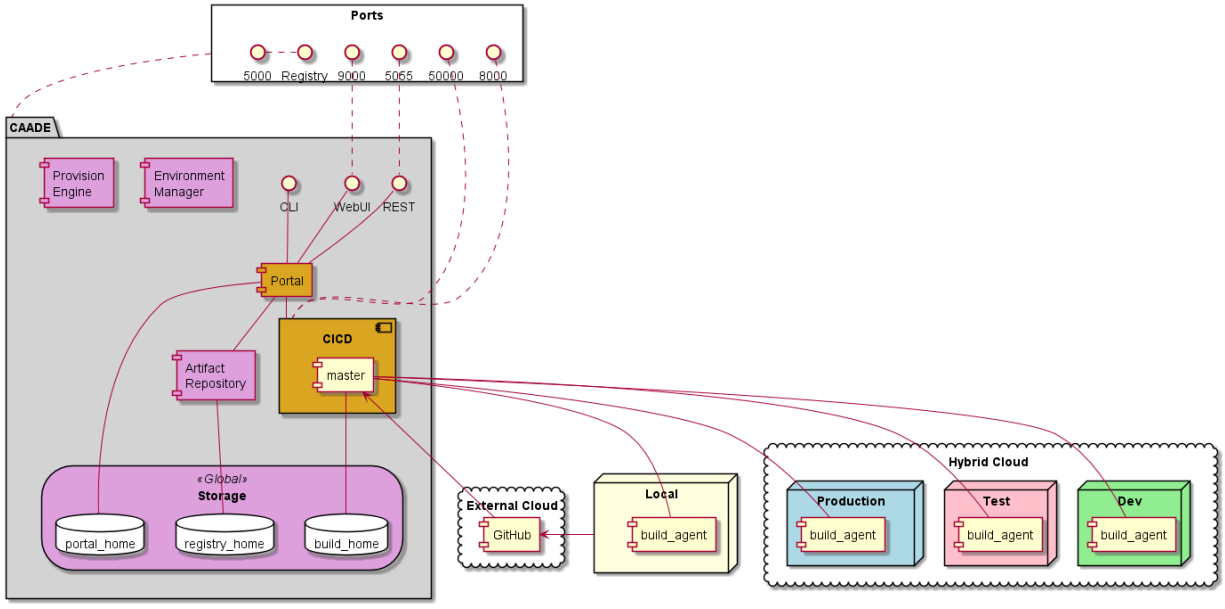
1.4 Process Architecture

This diagram shows how a developer interacts with CAADE to develop, test, and deploy cloud aware applications.


Developer
local
dev
test
production
create project
commit code
integrate code
build application
build status
failed
debug application
run unit tests
run integration tests
test results?
failed
debug application
bugs found
publish application
not
application deployed
deploy application
update application
start application

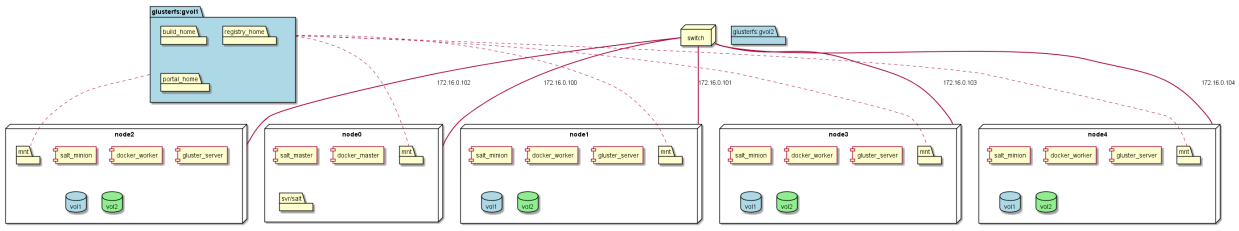
1.5 Deployment model

CAADE is made up a of a set of services and microservices to deliver capabilities to the :ref:'Actor-Developer'. The Service architecture shown in the deployment model is an example of an implementation of a CAADE architecture.



1.6 Physical Architecture

The physical architecture of CAADE is an example of a minimal hardware configuration that CAADE can be deployed.



CHAPTER 2

Solutions

2.1 Implementations

These are implementations of the architecture

2.1.1 CAADE Docker Solution

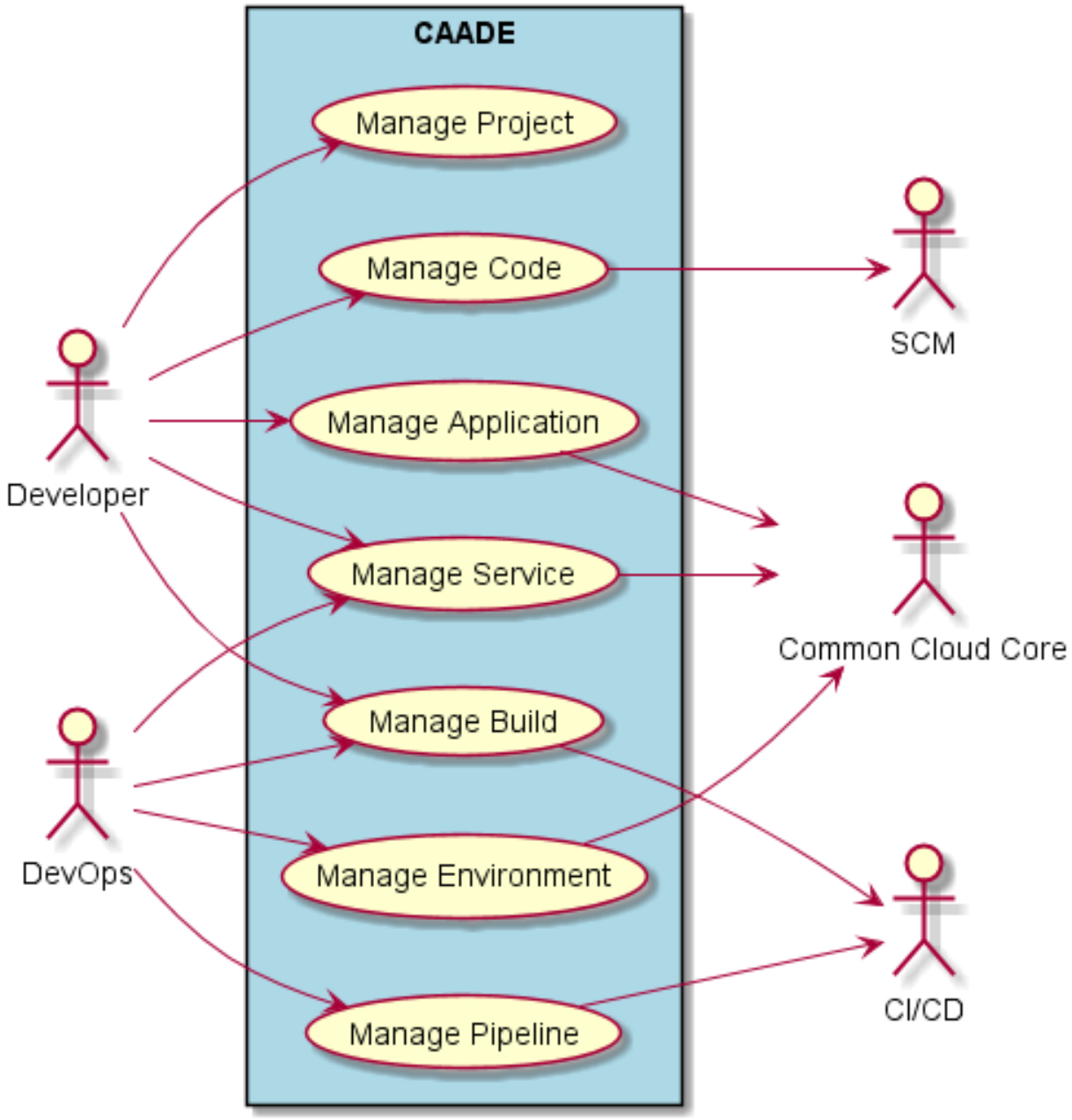
This solution is a Docker solution of the CAADE Architecture. It utilizes Docker Swarm, Jenkins, Salt Stack and GlusterFS. It is a simple example of the concepts of the CAADE architecture running on a small scale.

Actors

- *Developer*
- *DevOps*

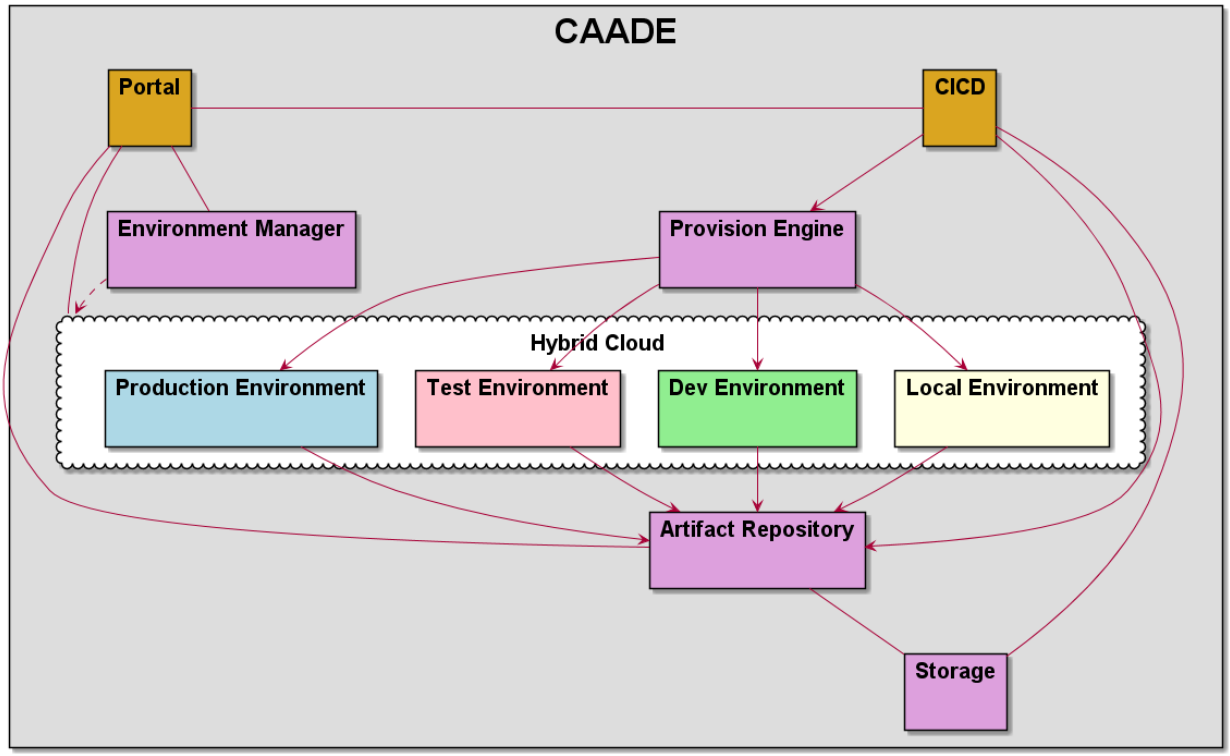
High level Use Cases

- *Manage Application*
- *Manage Build*
- *Manage Code*
- *Manage Environment*
- *Manage Pipeline*
- *Manage Project*
- *Manage Service*



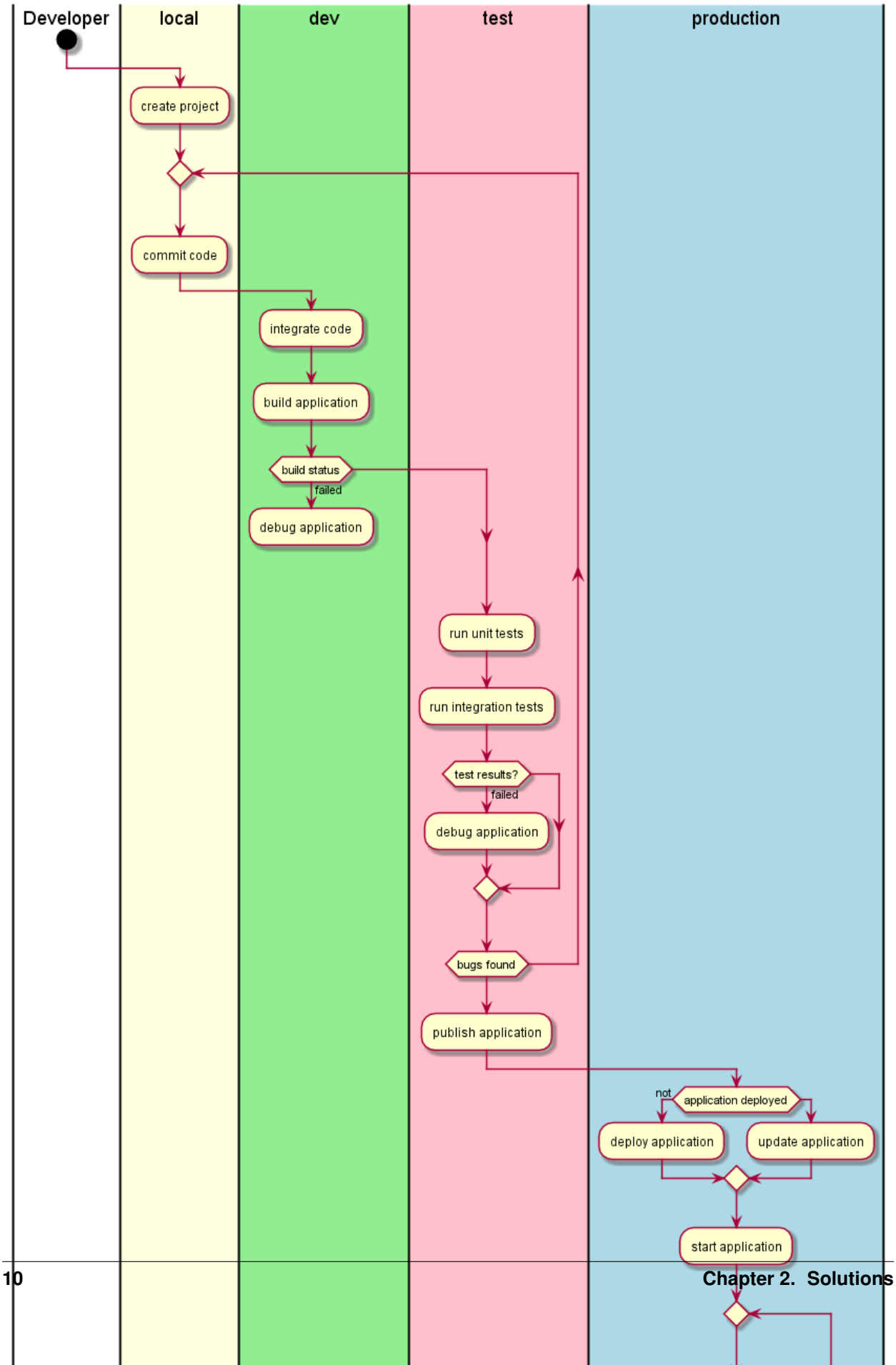
Logical Architecture

Developer need to focus on the development of applications. When code is modified and checked into a code repository like github. A *CICD* system will automatically build, test and deploy the application, microservice or project. Multiple environments that have been created in the Common Cloud Core will be used by CAADE and the *CICD* to promote applications across the different environments.



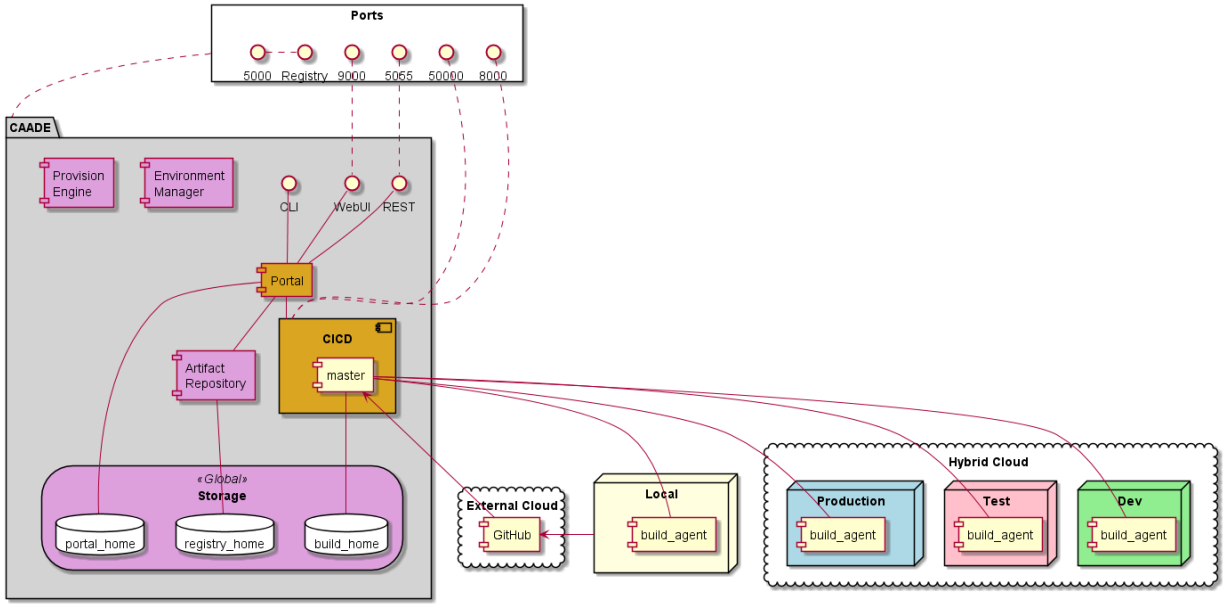
Process Architecture

This diagram shows how a developer interacts with CAADE to develop, test, and deploy cloud aware applications.


Developer
local
dev
test
production
create project
commit code
integrate code
build application
build status
failed
debug application
run unit tests
run integration tests
test results?
failed
debug application
bugs found
publish application
not
application deployed
deploy application
update application
start application

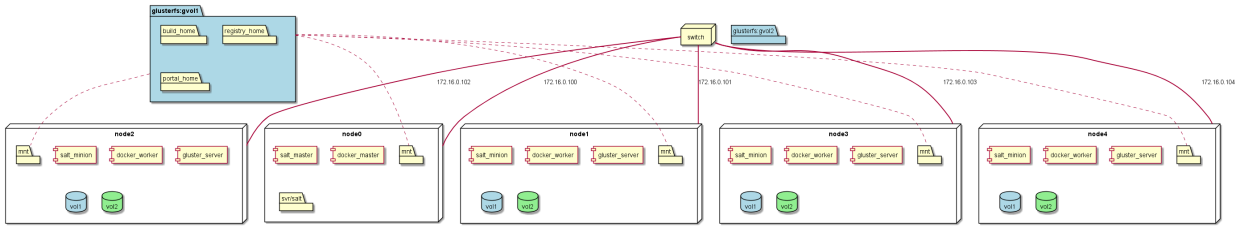
Deployment model

CAADE is made up a of a set of services and micro-services to deliver capabilities to the *Developer*. The Service architect shown in the deployment model is an example of an implementation of a CAADE architecture.



Physical Architecture

The physical architecture of CAADE is an example of a minimal hardware configuration that CAADE can be deployed.



Deployment

This is a Reference Architecture for the CAADE solution using Salt, Docker, Jenkins, and Gluster.

Salt Stack

Install Salt Master on Node 0

```
node0# sudo apt-get install salt-api
node0# sudo apt-get install salt-master
node0# sudo apt-get install salt-minion
```

Now that you have salt installed on node0 (master node). Go to the master configuration file /etc/salt/master and add these lines.

```
file_roots:
   base:
     - /srv/salt/
pillar_roots:
  base:
    - /srv/pillar
```

There should be several things that are in the /etc/salt/master file commented out.

Get the fingerprint of the master node

```
node0# sudo salt-key -f master.pub
```

Save this string it will be used in the configuration of the minions.

Install Salt Minion on Node[0-4]

```
node1# sudo apt-get install salt-minion
```

Now edit the /etc/salt/minion file to contain the following

```
master: node0
master_finger: "Put output of 'alt-key -f master.pub' here"
```

Get things running

On node0 start the salt master as root in the foreground

```
node0# sudo salt-master
```

or in the background

```
node0# sudo salt-master -d
```

On node[0-4] start the salt-minions

```
node1# sudo salt-minion
```

or in the background with the -d flag

```
node1# sudo salt-minion -d
```

now go back to node0 and accept the minions into the salt stack

```
node0# sudo salt-key -A
```

Now you can test and see if salt can see all of the nodes

```
node0# salt "*" test.ping
node0:
    True
node1:
    True
node2:
    True
node3:
```

(continues on next page)

(continued from previous page)

```
    True
node4:
    True
```

1. Configure Salt states
2. Configure Salt Pillar
3. Download Salt Formula for CAADE

Install Gluster

Install Gluster on each of the nodes (node[0-4])

```
node0# sudo apt-get update
```

Install GlusterFS package using the following command

```
node0# sudo apt-get install -y glusterfs-server
```

Start the glusterfs-server service on all gluster nodes.

```
node0# sudo service glusterfs-server start
```

Create Volumes for Gluster to use This assumes that you already have drives that have been mounted.

```
sudo mkdir -p /data/gluster
sudo mount /dev/sdb1 /data/gluster
```

Add an entry to /etc/fstab for keeping the mount persistent across reboot.

```
echo "/dev/sdb1 /data/gluster ext4 defaults 0 0" | sudo tee --append /etc/fstab
```

Now attach all of the nodes to each other. Go to node0 and type the following.

```
node0# sudo gluster peer probe node1
node0# sudo gluster peer probe node2
node0# sudo gluster peer probe node3
node0# sudo gluster peer probe node4
```

Now you can add volumes to the gluster cluster

```
node0# salt "*" cmd.run "mkdir -d /data/gluster/gvol0"
node0# sudo gluster volume create gvol0 replica 2 node1:/data/gluster/gvol0 node2:/
↪data/gluster/gvol0
node0# sudo gluster volume start gvol0
node0# sudo gluster volume info gvol0
```

Mount Gluster Volumes on all of the nodes

Now you have created a volume and now you can access it on all of the nodes by mounting it.

```
node0# mkdir /mnt/glusterfs
node0# mount -t glusterfs node1:/gvol0 /mnt/glusterfs
```

To make the mount permanent across reboots you need to add it to the fstab

```
node0# echo "node1:/gvol0 /mnt/glusterfs glusterfs  defaults,_netdev 0 0" | echo tee -
↪-append /etc/fstab
```

Additional information can be found [here](http://www.itzgeek.com/how-tos/linux/ubuntu-how-tos/install-and-configure-glusterfs-on-ubuntu-16-04-debian-8.html)

Docker Swarm

There is a great blog on how to generically set this up [here](http://btmiller.com/2016/11/27/docker-swarm-1.12-cluster-orchestration-with-saltstack.html).

1. Using Salt Stack install
2. Test Docker Swarm Installation

Jenkins Configuration

Install the following plugins 1. Self-Organizing Swarm Plug-in Modules

Registry Configuration

An RSA key is needed for the local Docker Registry. This can be done with OpenSSL. The docker-compose.yaml file for the deployment of CAADE stores in volumes that are monunted into the containter. The domain.key and domain.cert files should be accessible.

Generating the openssl certs.

So we need to generate the key and cert in a ./registry_certs directory in the same path of where you run the stack deploy command. so you will need to create a directory named registry_certs and then run the openssl command.

```
# mkdir registry_certs
# openssl req -newkey rsa:4096 -nodes -sha256 -keyout registry_certs/domain.key \
 -x509 -days 356 -out registry_certs/domain.cert
Generating a 4096 bit RSA private key
.....................++
.....................................................................................
↪.....................................................++
writing new private key to 'registry_certs/domain.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]: CA
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []: <registry server>
Email Address []:
```

Now that the keys are created and put into a directory that the docker compose file references and put into the secrets docker will now know where to find the keys and certs to launch a secure private repository. Make sure that you access the registry through the same name specified in the key generation <registry server>. Ideally this would be a common name that all clients can access through exposure of the ports through the docker stack deploy. Such as the docker swarm node machine.

Putting the certs in each Client

Using the CAADE configuration we put the domain.cert in the /etc/docker/certs.d directory. Specifically we need to put the domain.cert into the /etc/docker/certs.d/<registry_address>/ca.cert. In order to do this simply we need to put the domain.cert in a mounted filesystem and using salt to update docker client. This will make it so every node in the docker swarm can access the local private repository.

```
cp -r ./registry-certs /mnt/registry/registry-certs
salt "*" cmd.run "mkdir -p /etc/docker/certs.d/node0:5000"
salt "*" cmd.run "cp /mnt/registry/registry-certs/domain.cert /etc/docker/certs.d/
↪node0:5000/ca.crt"
salt "*" cmd.run "service docker restart"
```

Deploying the stack of services

Now that the environment is set up. You can now deploy the stack to your cluster of machines. You will need to:

1. Get the latest release from github.
2. Copy the registry_certs to the deploy directory. For the local private Registry.
3. Deploy the stack to docker.

```
# git clone http://github.com/CAADE/CAADE
# cd deploy
# cp -r <registry_certs> ./registry_certs
# docker stack deploy --compose-file production/docker-compose.yaml caade
```

2.2 Services

These are the micro-services of the c3 Solution that are used to implement the solutions.

2.2.1 Services

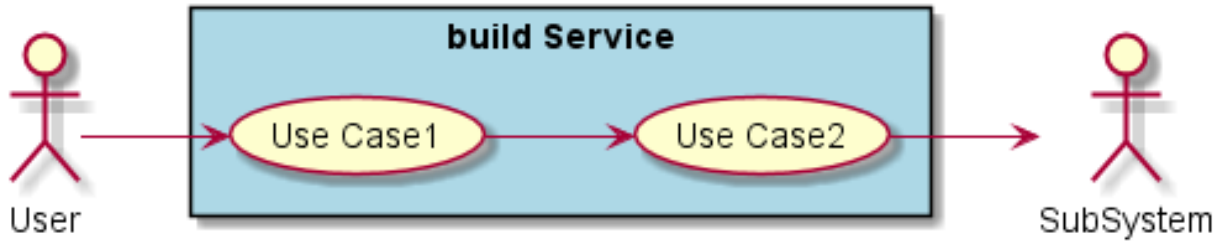
The system is implemented using micro-services that are deployed across a cloudified architecture.

build

build is a micro-service of caade . . .

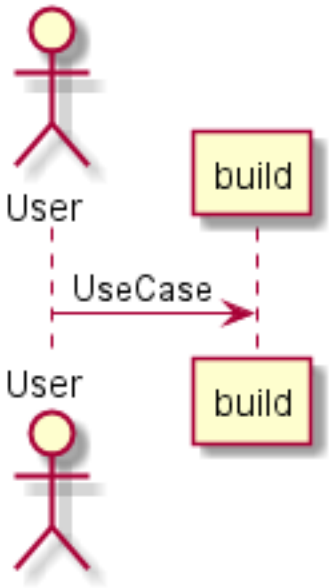
Use Cases

-



Users

- *DevOps*



Uses

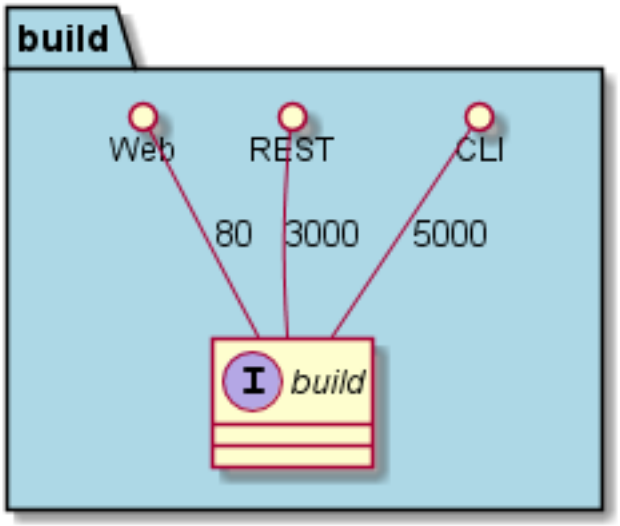
- *build*

Interface

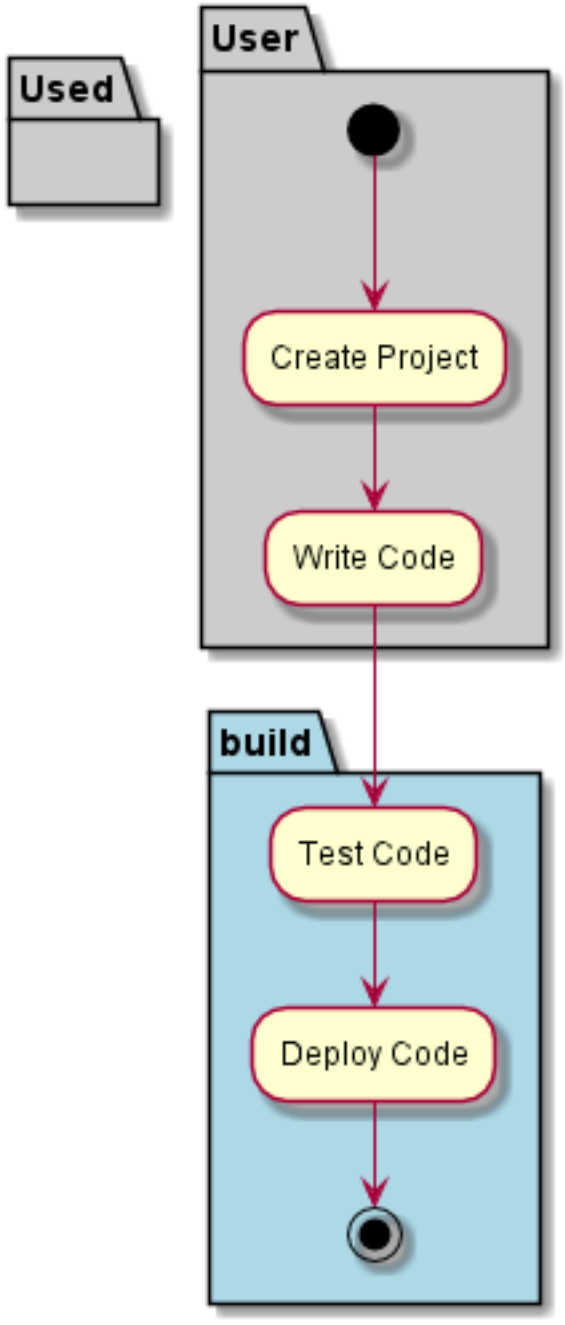
- CLI - Command Line Interface
- REST-API -
- Portal - Web Portal

Logical Artifacts

-

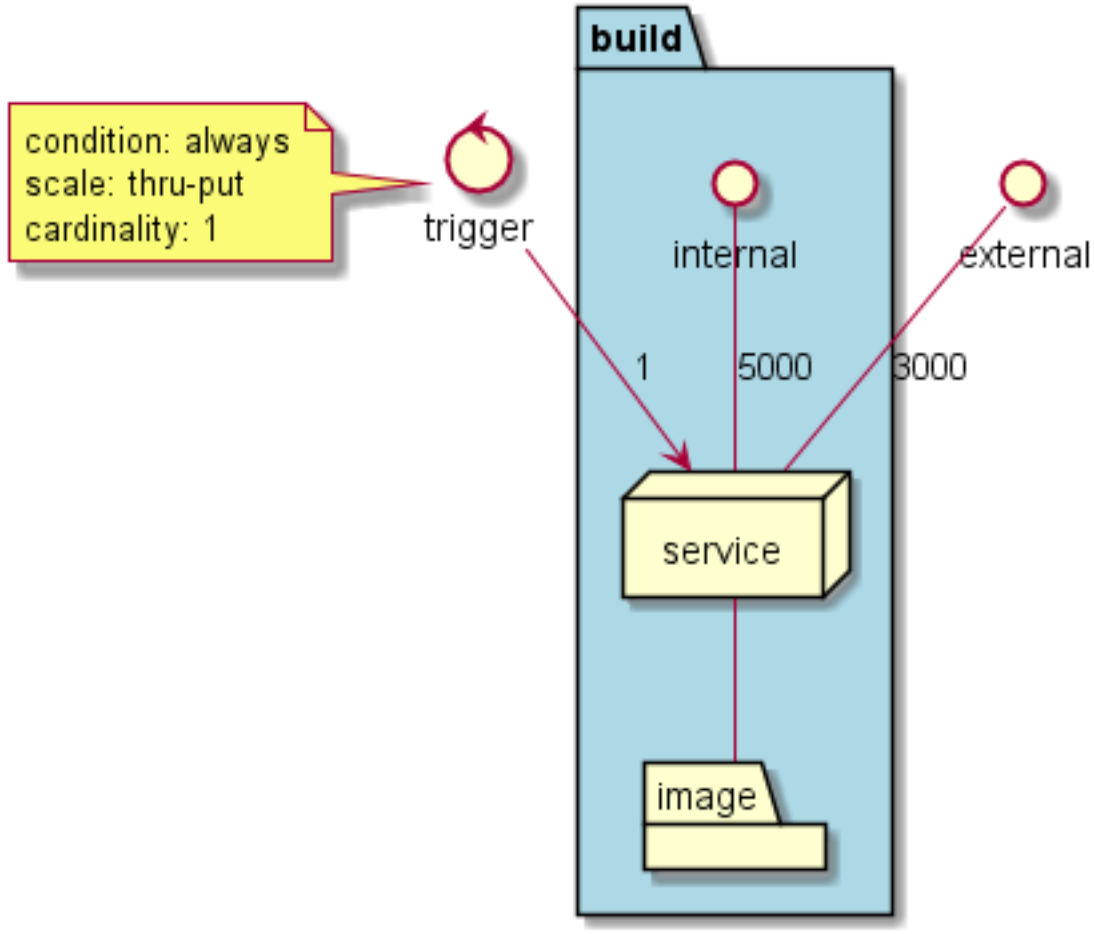

build
Web
REST
CLI
80
3000
5000
build

Activities and Flows



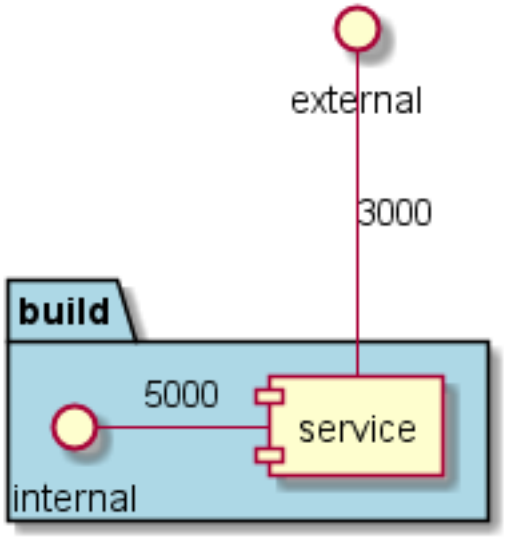
Deployment Architecture

This is the deployment of the micro-service. The micro-service is deployed when *trigger* and should scale from # *to* # based on *condition*. The micro-service is deployed with the *imagename* image. The ports exposed are 5000 for external and 3000 for internal.



Physical Architecture

The micro-services are physically deployed on to a hybrid cloud infrastructure.

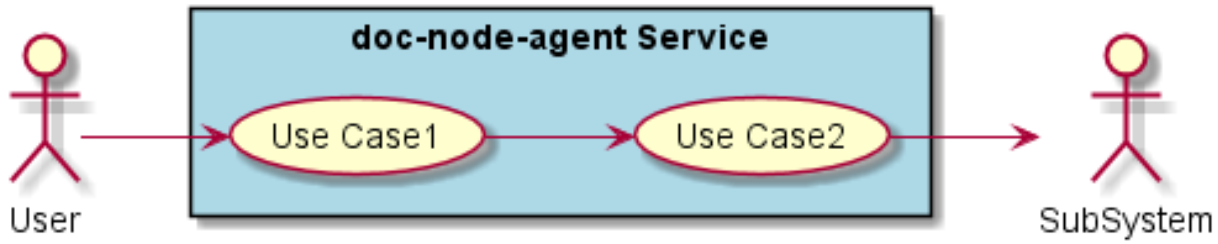


doc-node-agent

doc-node-agent is a micro-service of caade . . .

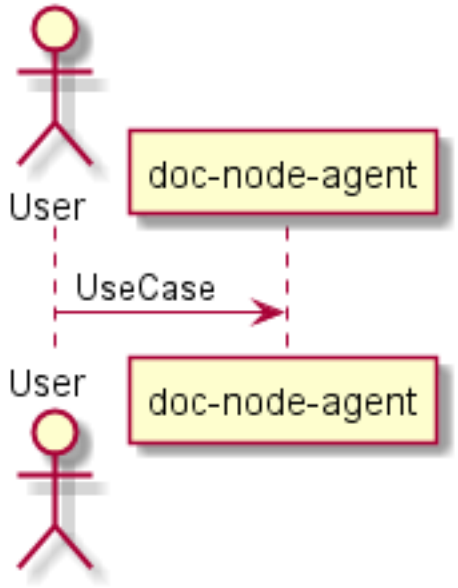
Use Cases

-



Users

- *DevOps*



Uses

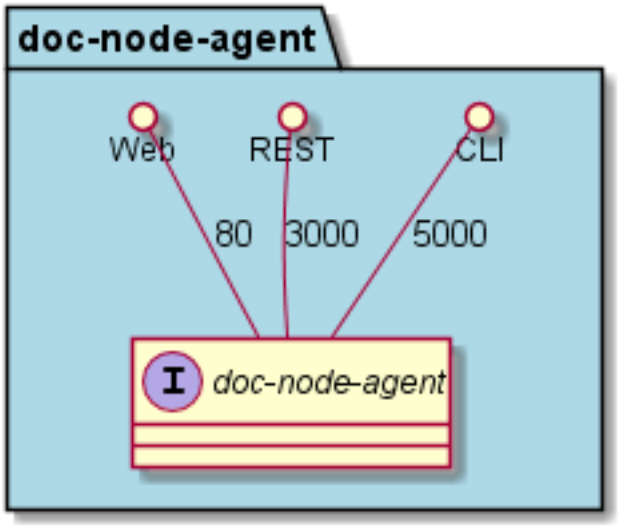
- *doc-node-agent*

Interface

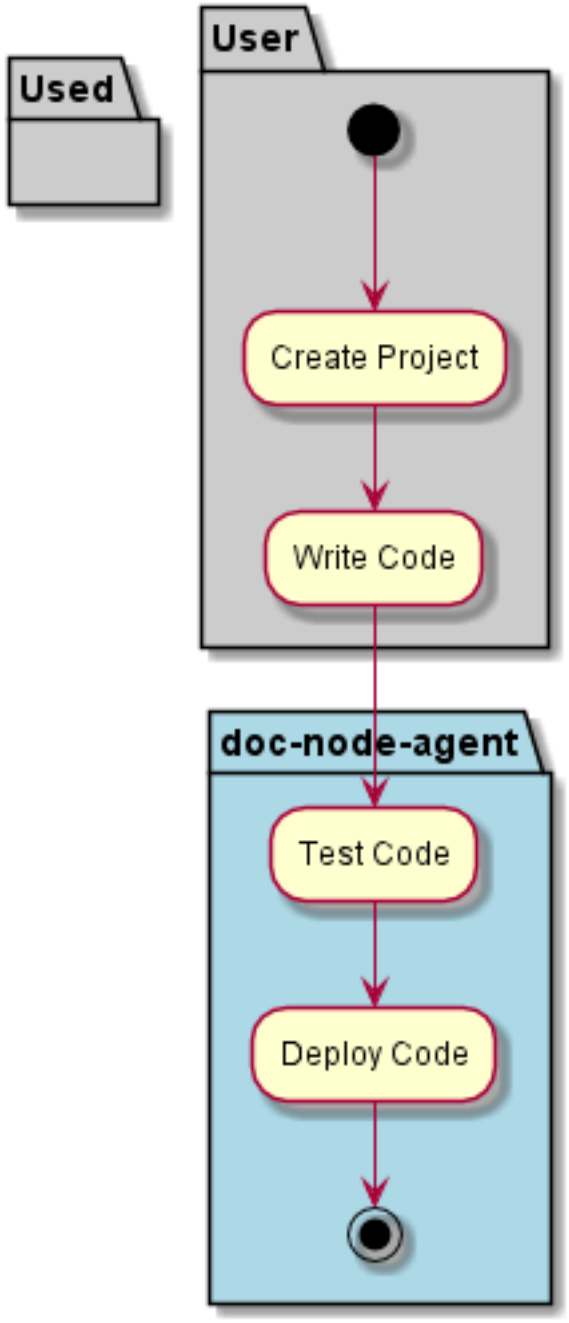
- CLI - Command Line Interface
- REST-API -
- Portal - Web Portal

Logical Artifacts

-

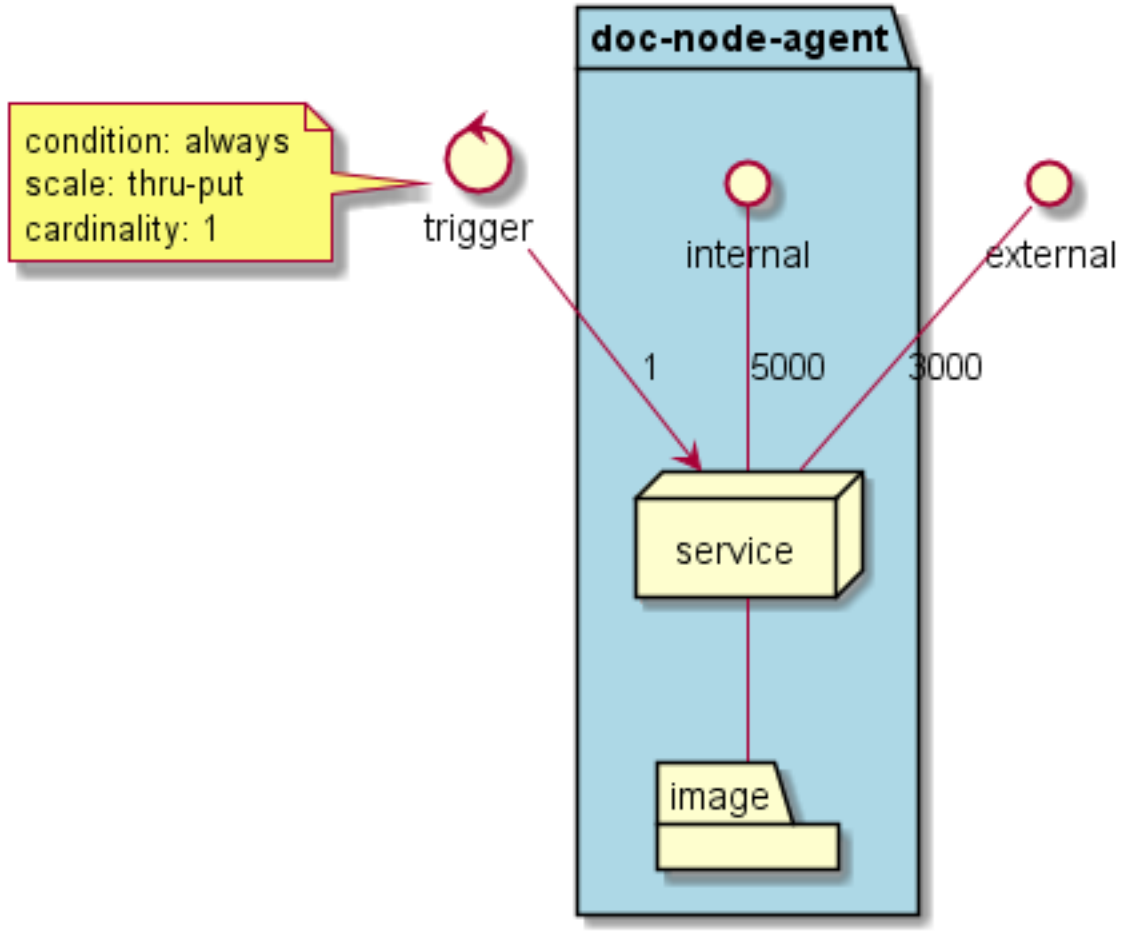

doc-node-agent
Web
REST
CLI
80
3000
5000
doc-node-agent

Activities and Flows



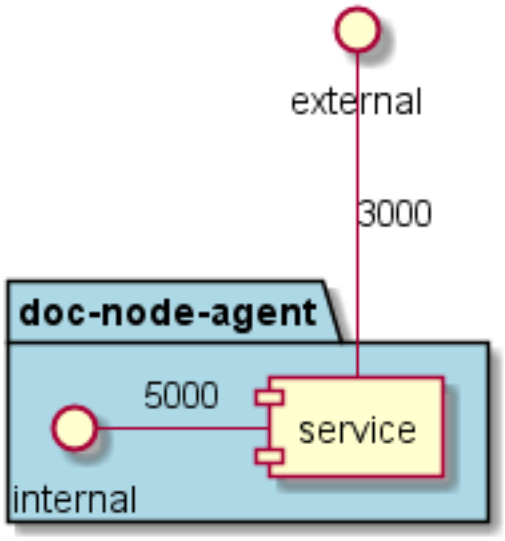
Deployment Architecture

This is the deployment of the micro-service. The micro-service is deployed when *trigger* and should scale from # *to* # based on *condition*. The micro-service is deployed with the *imagename* image. The ports exposed are 5000 for external and 3000 for internal.



Physical Architecture

The micro-services are physically deployed on to a hybrid cloud infrastructure.

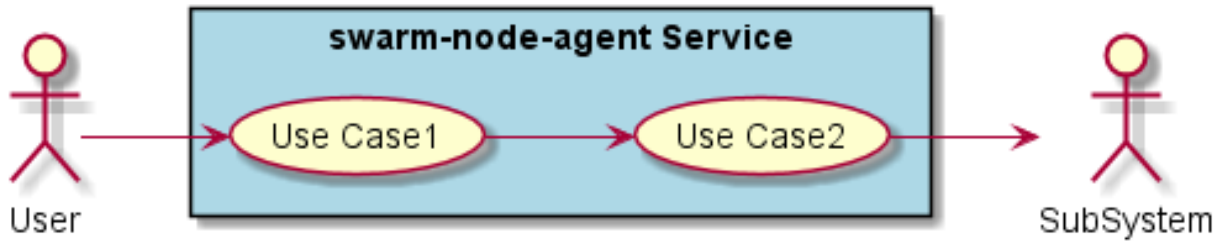


swarm-node-agent

swarm-node-agent is a micro-service of caade . . .

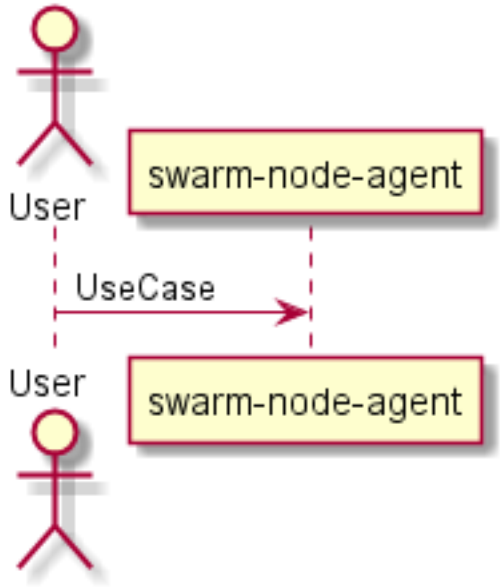
Use Cases

-



Users

- *DevOps*



Uses

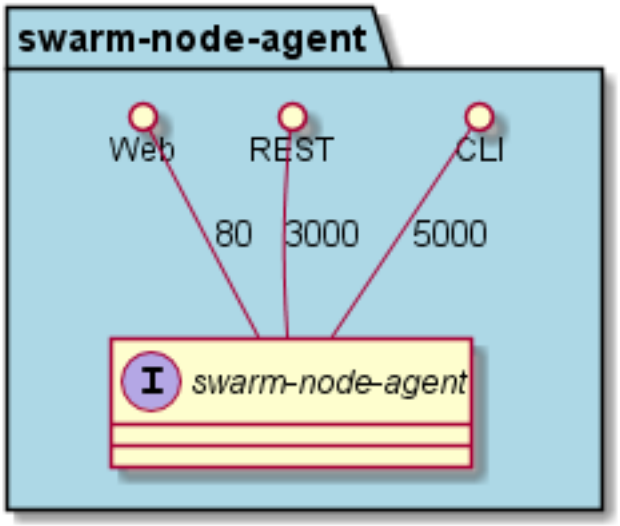
- *swarm-node-agent*

Interface

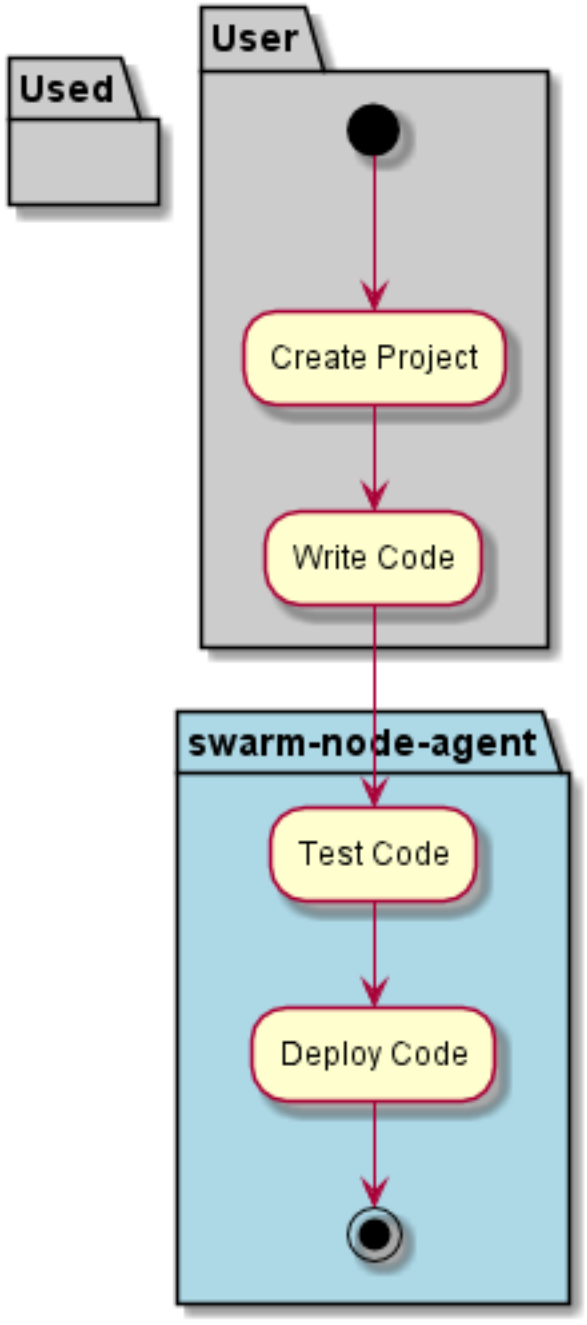
- CLI - Command Line Interface
- REST-API -
- Portal - Web Portal

Logical Artifacts

-

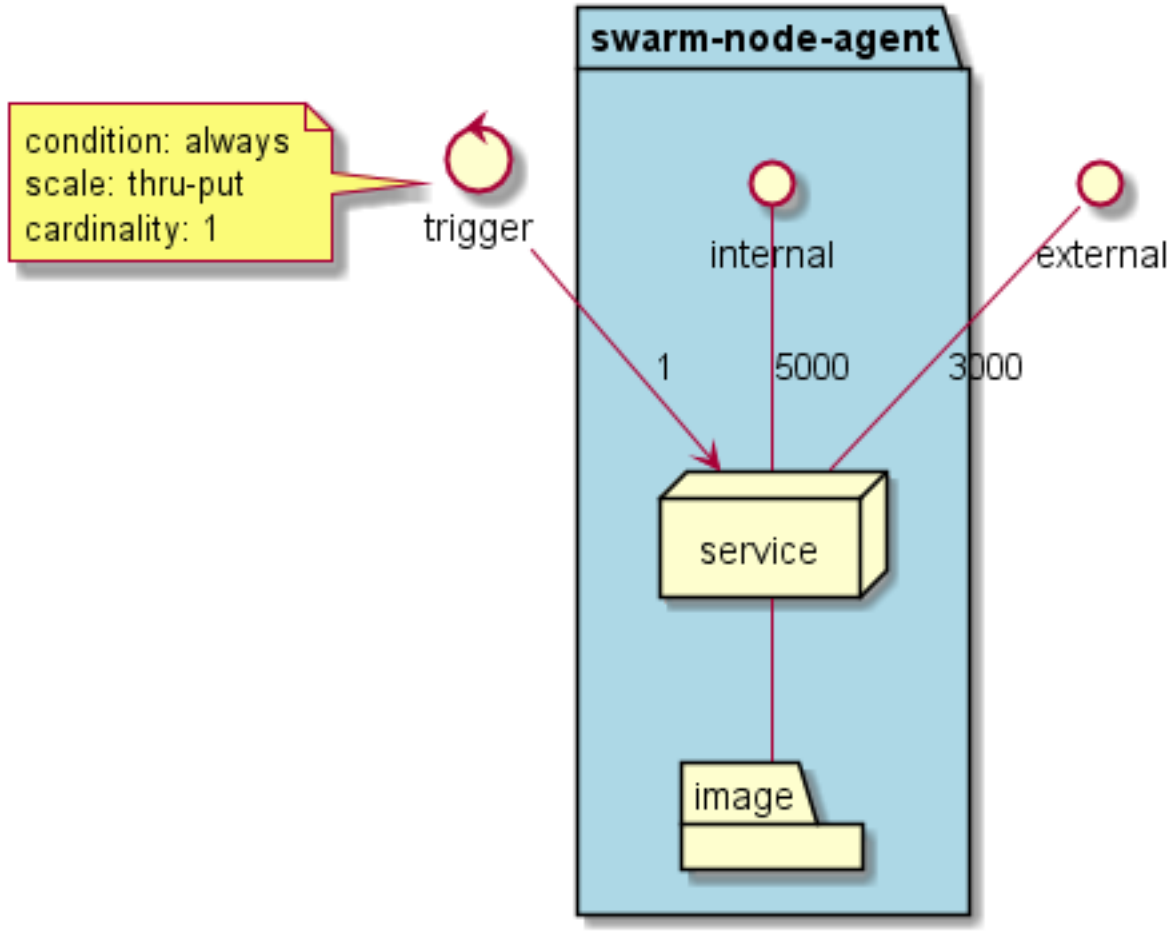

swarm-node-agent
Web
REST
CLI
80
3000
5000
I
swarm-node-agent

Activities and Flows



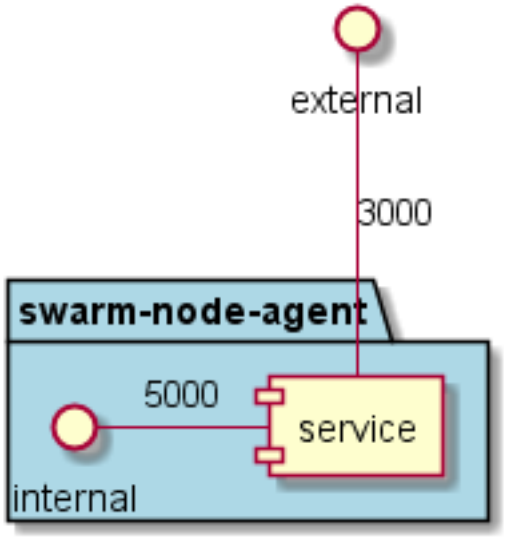
Deployment Architecture

This is the deployment of the micro-service. The micro-service is deployed when *trigger* and should scale from # *to* # based on *condition*. The micro-service is deployed with the *imagename* image. The ports exposed are 5000 for external and 3000 for internal.



Physical Architecture

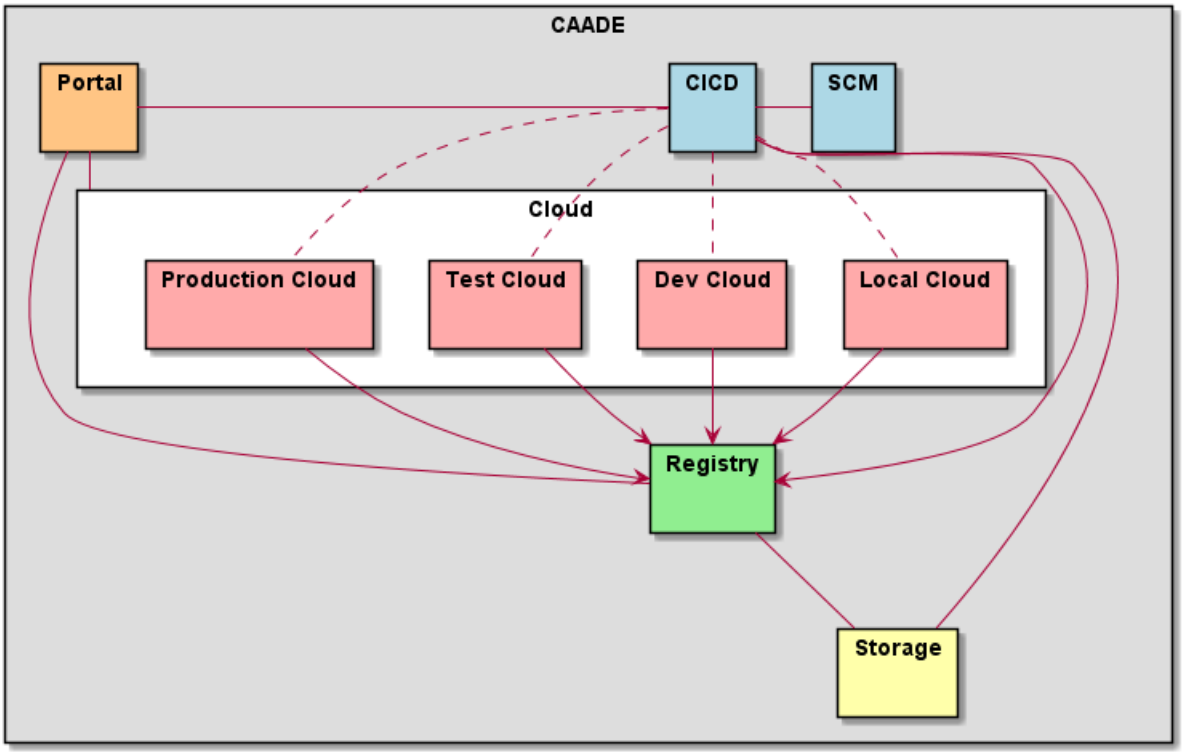
The micro-services are physically deployed on to a hybrid cloud infrastructure.



2.3 Sub Systems

These are the high level Subsystems of the c3 Solution

- *Automation Framework*
- *CICD*
- *Hybrid Cloud*
- *Hybrid Cloud / Artifact Repository*
- *Hybrid Cloud/Environment Manager*
- *Hybrid Cloud/Provision Engine*
- *Hybrid Cloud/Storage*
- *SCM*

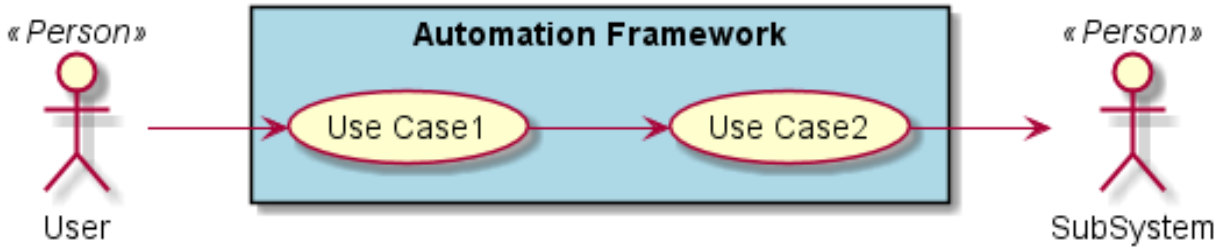


2.3.1 Automation Framework

Automation Framework is a subsystem of caade . . .

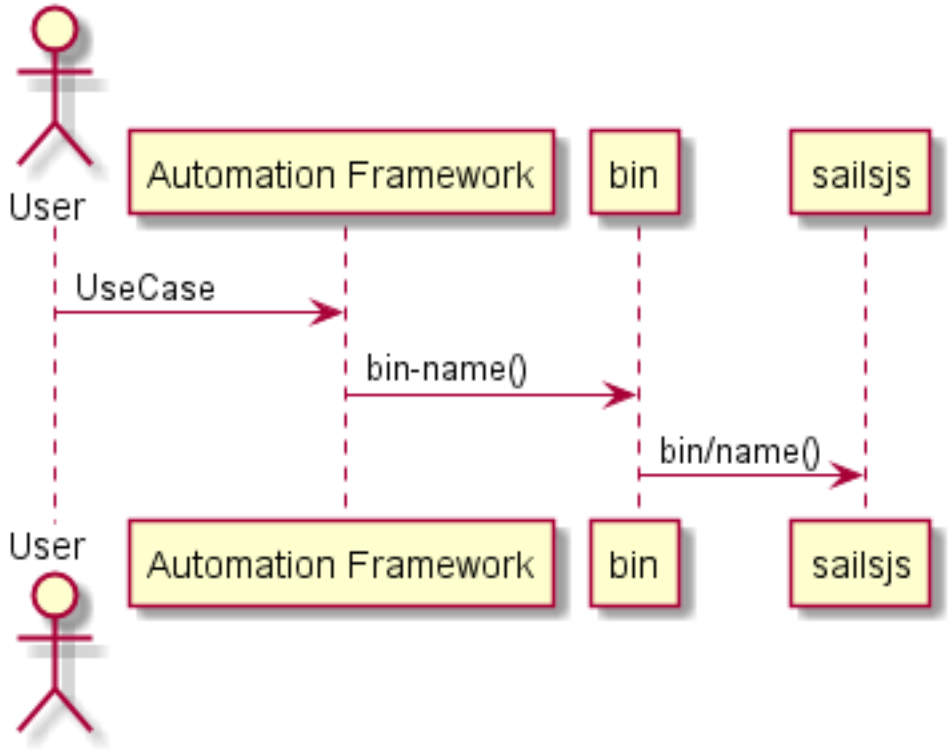
Use Cases

-



Users

- *DevOps*



Uses

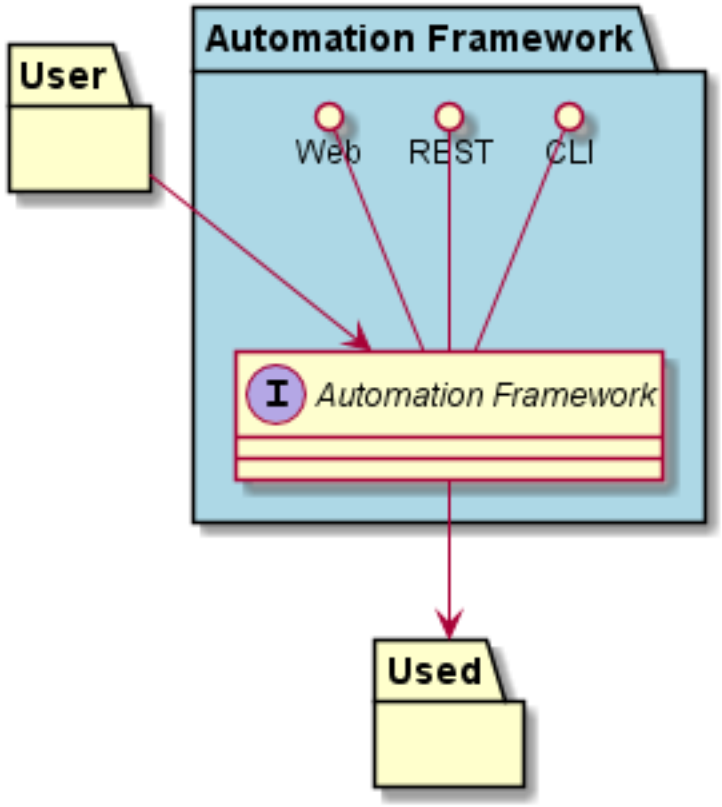
- *Automation Framework*

Interface

- CLI - Command Line Interface
- REST-API -
- Portal - Web Portal

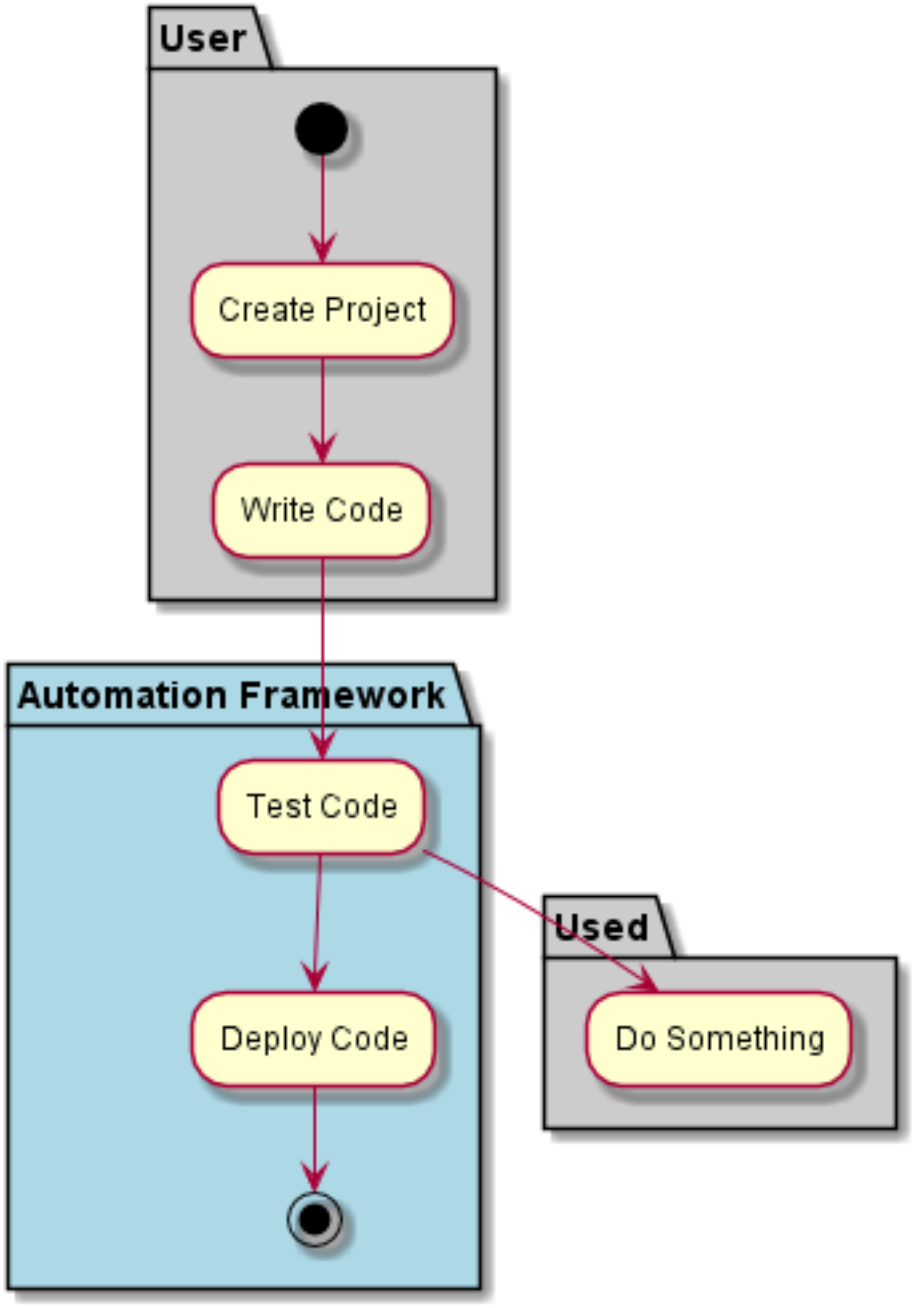
Logical Artifacts

-



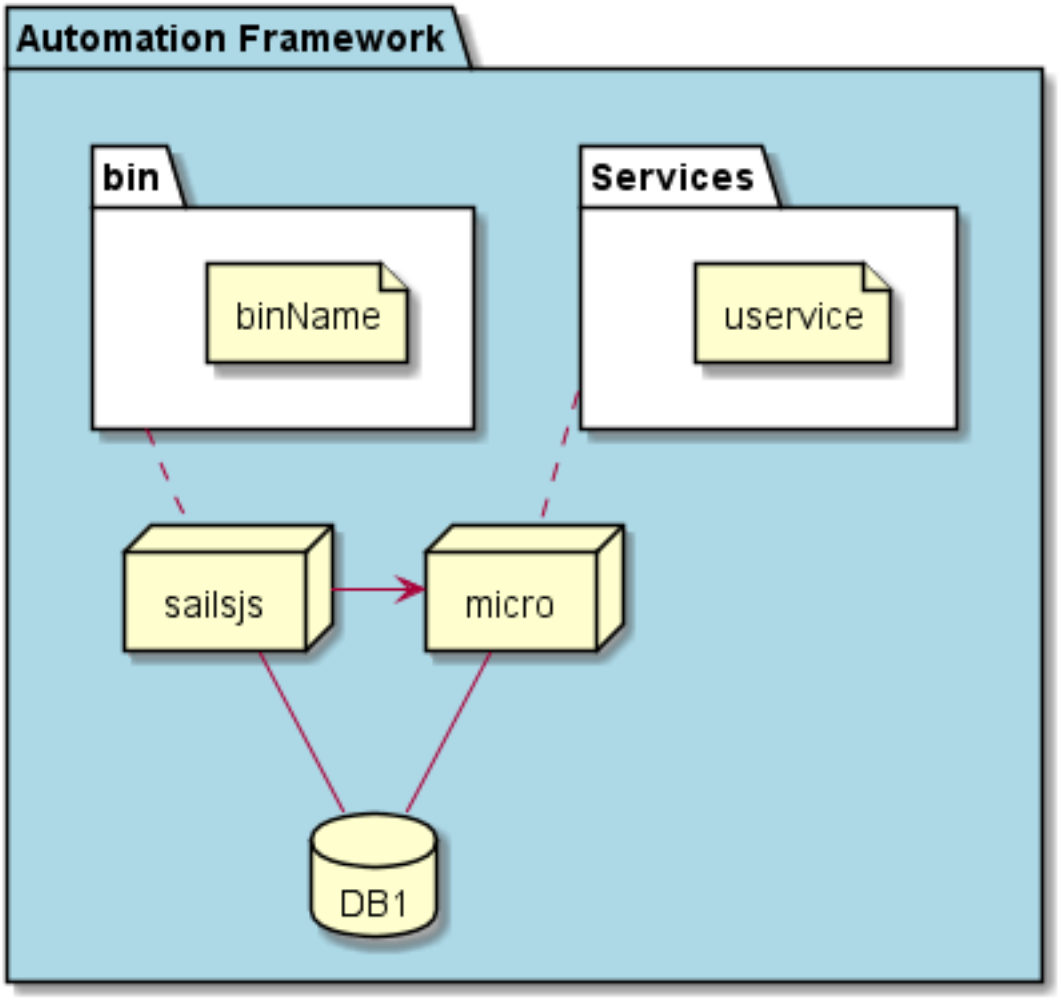
Activities and Flows

The Automation Framework subsystem provides the following activities and flows.



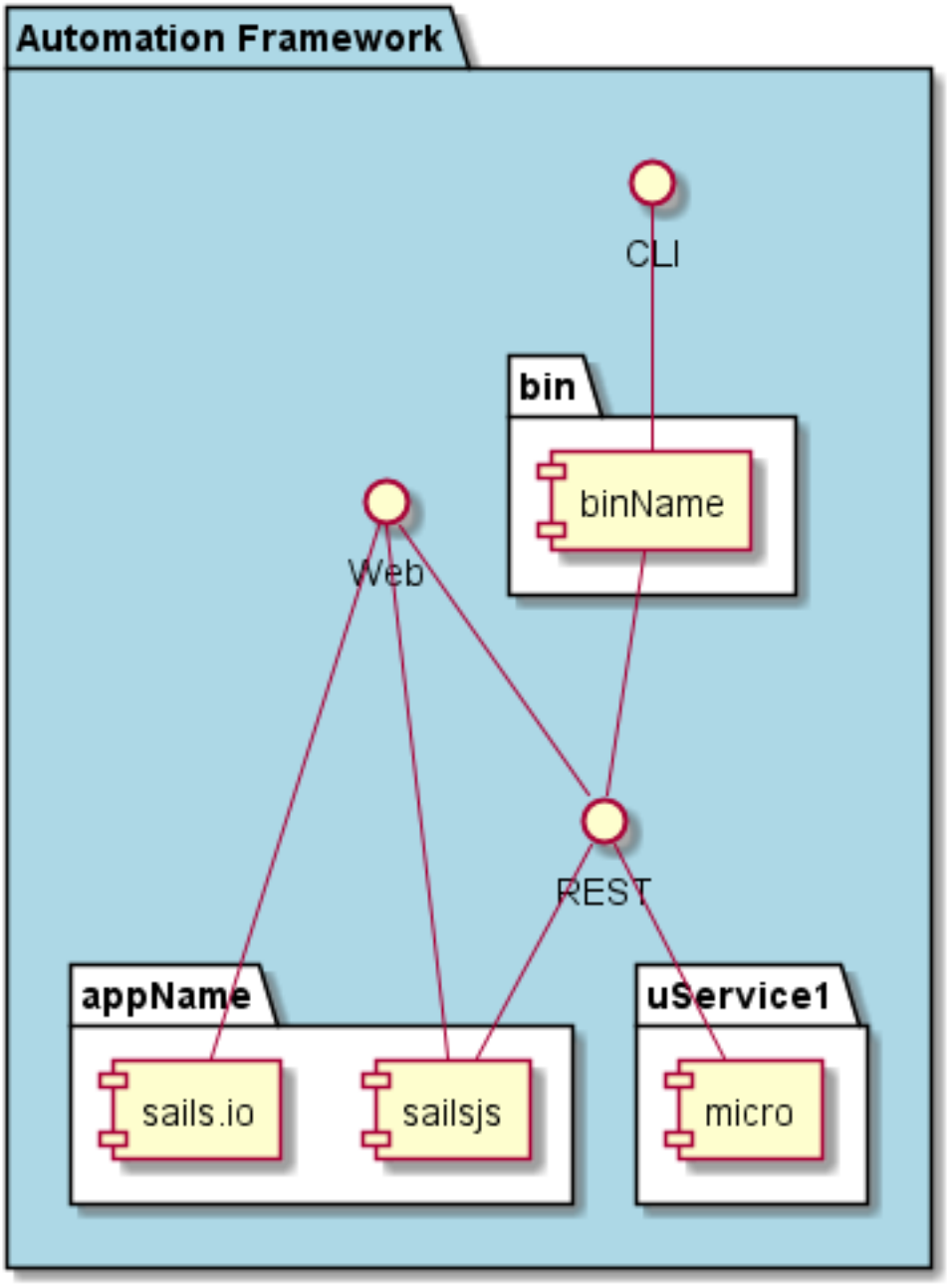
Deployment Architecture

This subsystem is deployed using micro-services as shown in the diagram below. The 'micro' module is used to implement the micro-services in the system. The subsystem also has an CLI, REST and Web Interface exposed through a sailajs application. The sailsjs application will interface with the micro-services and can monitor and drive work-flows through the mesh of micro-services.



Physical Architecture

The Automation Framework subsystem is is physically laid out on a hybrid cloud infrastructure. Each microservice is shown how they connect to each other. All of the micro-services communicate to each other and the main app through a REST interface. A CLI, REST or Web interface for the app is how other subsystems or actors interact. Requests are forwarded to micro-services through the REST interface of each micro-service.



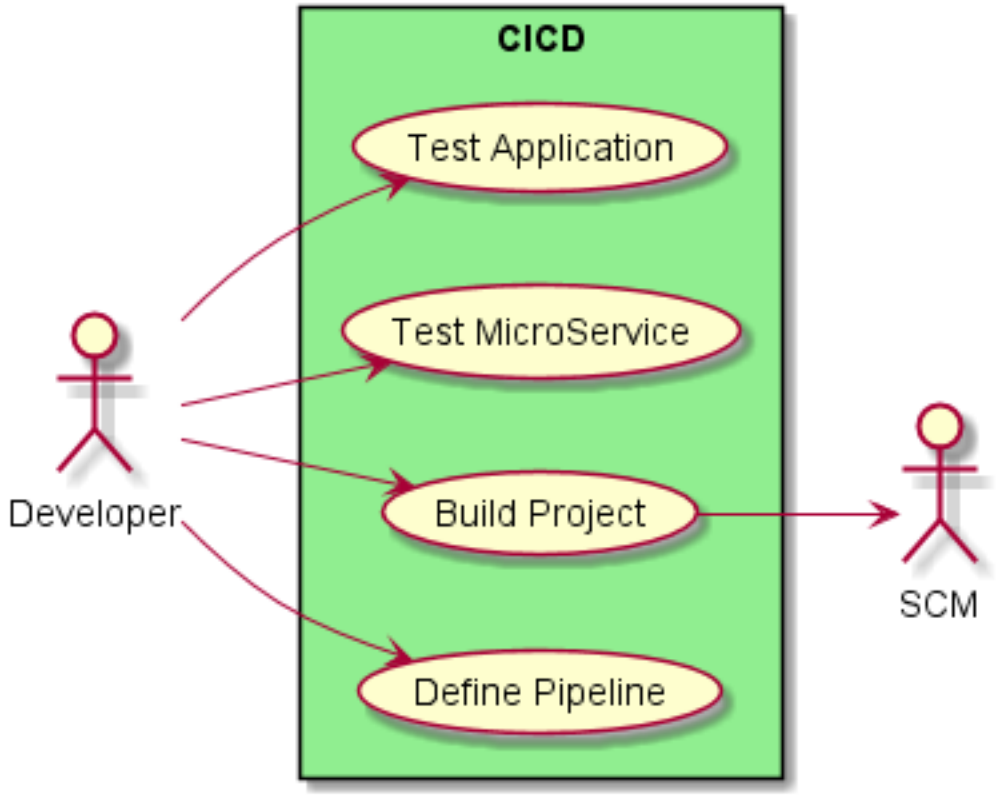
Micro-Services

2.3.2 CICD

CICD is a subsystem of caade that is implemented by an existing CI/CD service that is available today. Examples of CICD systems that can be used are Jenkins, Bamboo, TravisCI, etc…

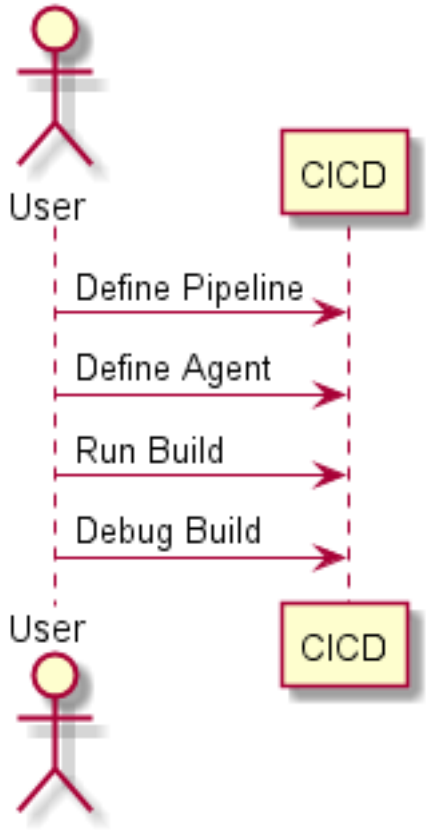
Use Cases

- *Test Application*



Users

- *Developer*



Uses

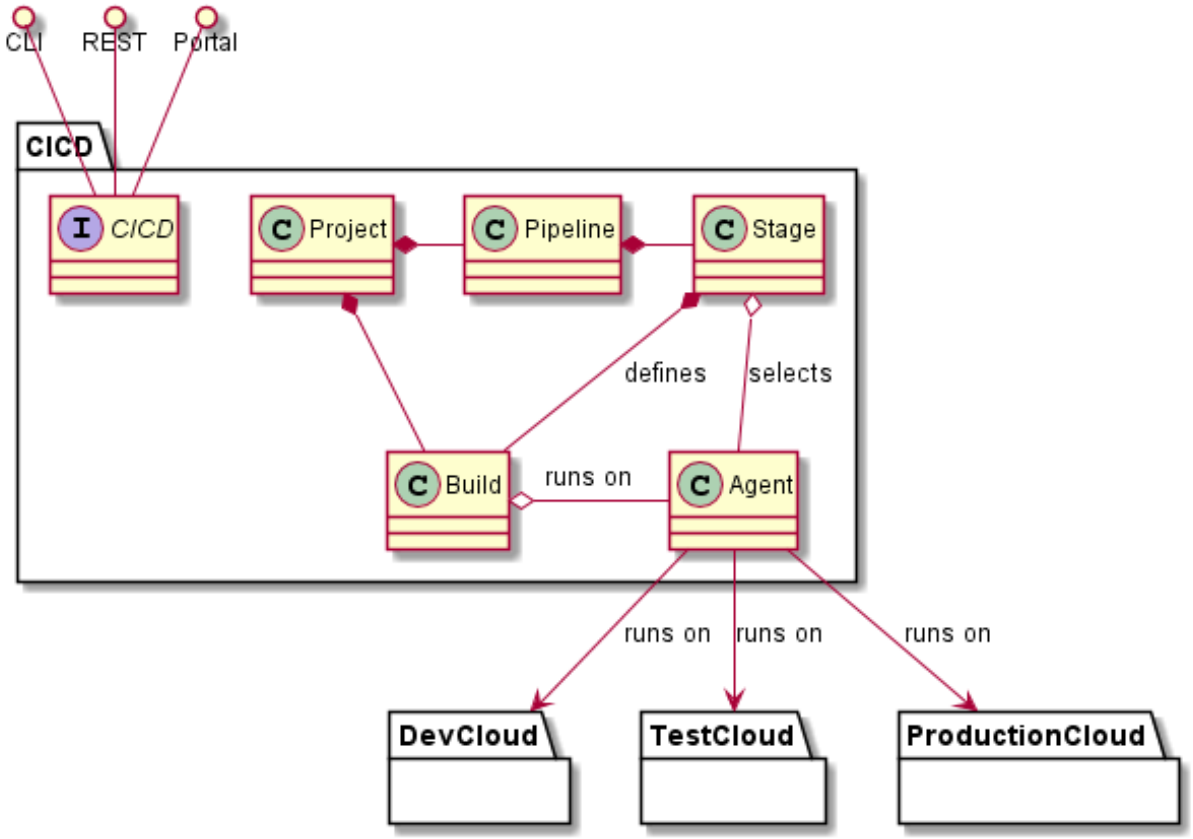
- *CICD*

Interface

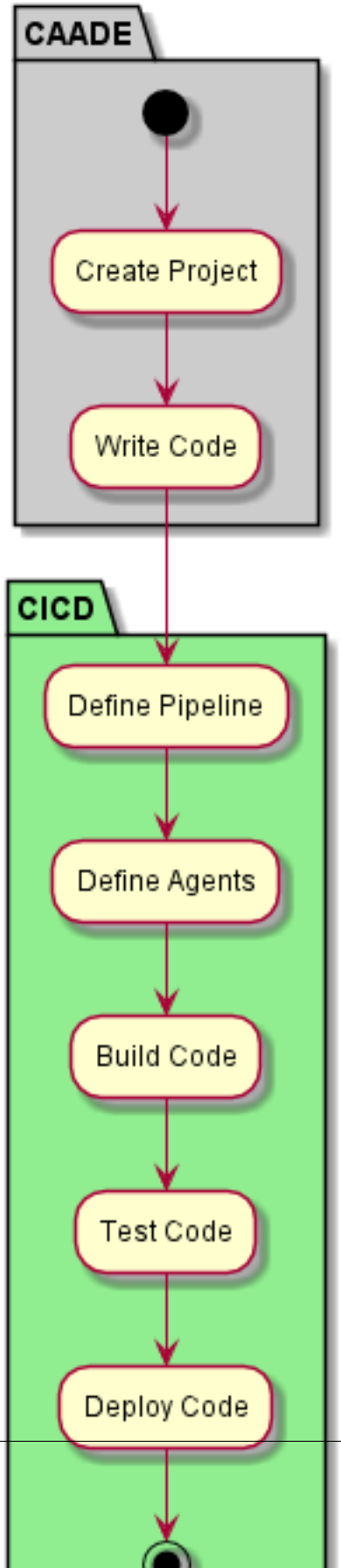
- CLI - Command Line Interface
- REST-API
- Portal - Web Portal

Logical Artifacts

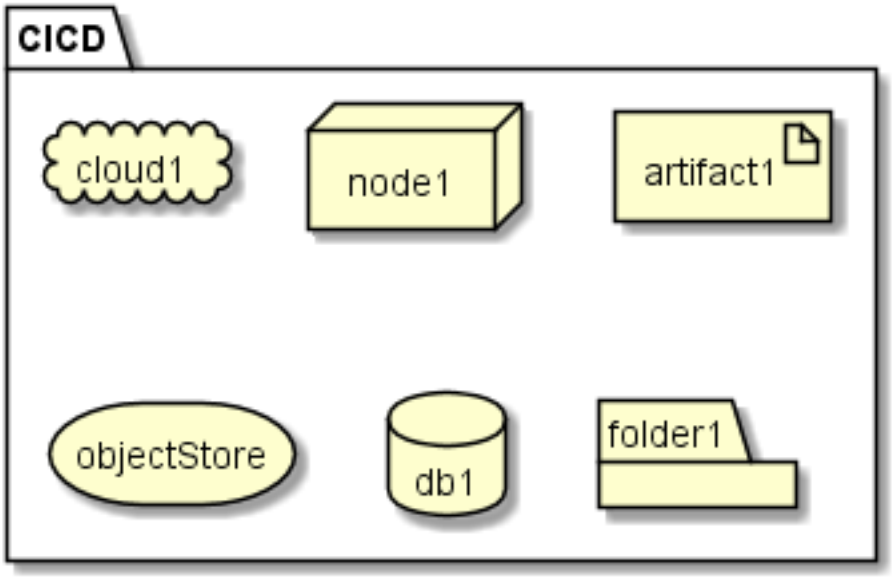
- Agent - Agent running in the different clouds that perform builds for a Project
- Build - Build Stages of a pipeline for a project.
- Pipeline - Pipeline that defineds how a project is built, test, and deployed
- Project - Project that contains the application and microservices
- Stage - Stage of builds defined in the pipeline.


CLI
REST
Portal
CICD
CICD
Project
Pipeline
Stage
defines
selects
Build
runs on
Agent
runs on
runs on
runs on
DevCloud
TestCloud
ProductionCloud

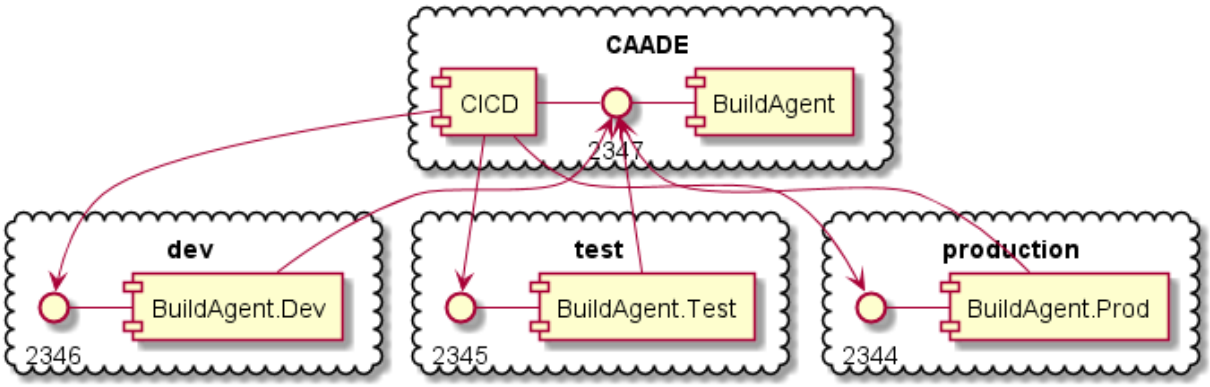
Activities and Flows



Deployment Architecture



Physical Architecture

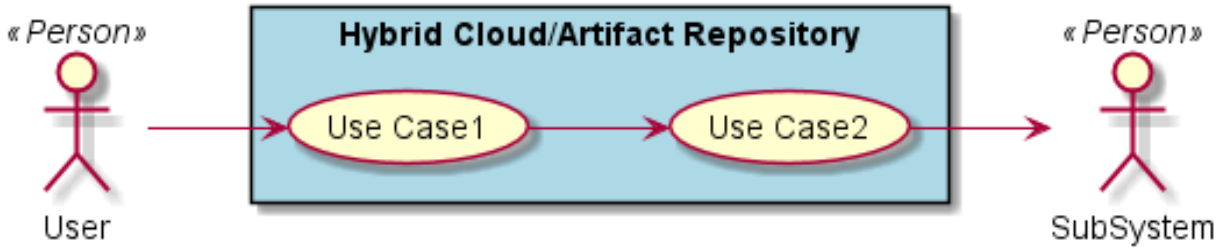


2.3.3 Hybrid Cloud / Artifact Repository

Hybrid Cloud/Artifact Repository is a subsystem of caade . . .

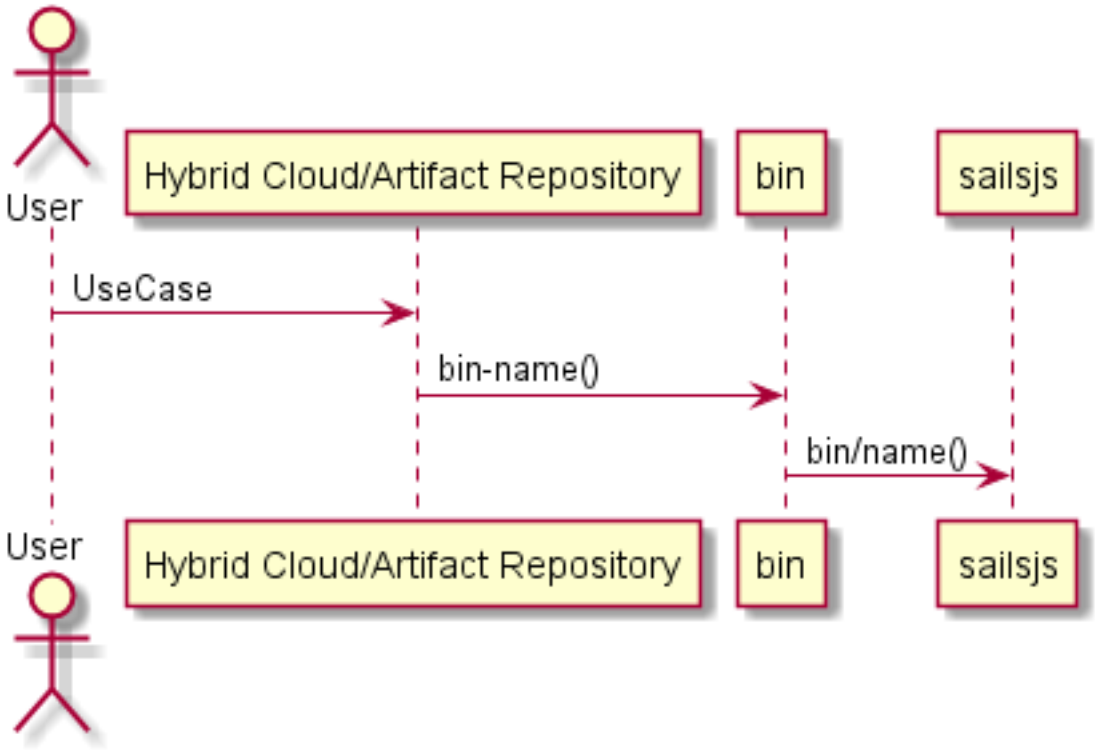
Use Cases

-



Users

- *DevOps*



Uses

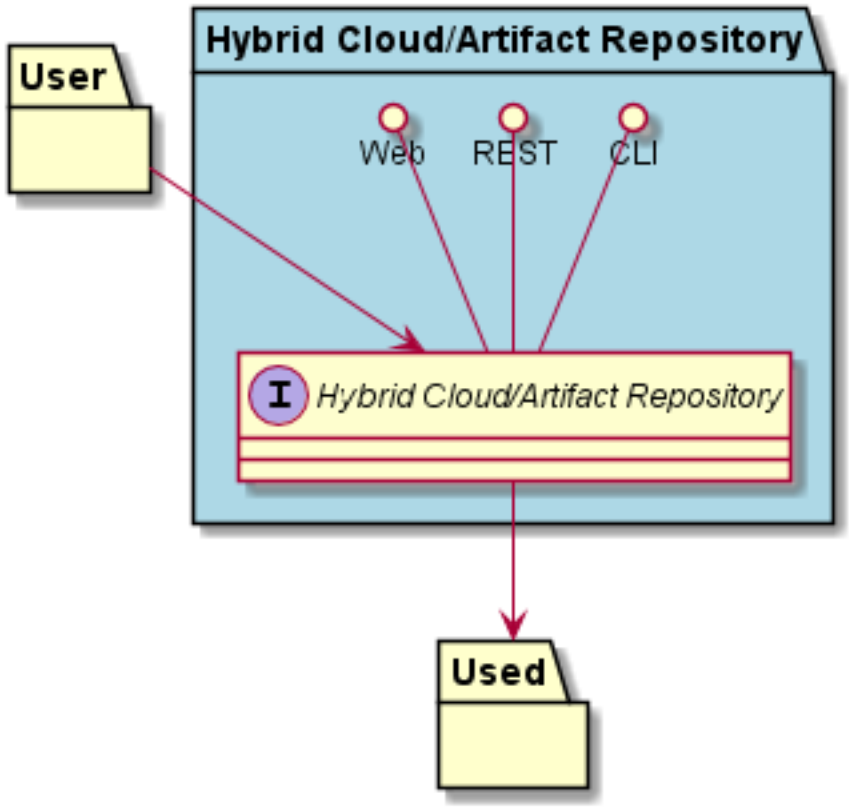
- *Hybrid Cloud / Artifact Repository*

Interface

- CLI - Command Line Interface
- REST-API -
- Portal - Web Portal

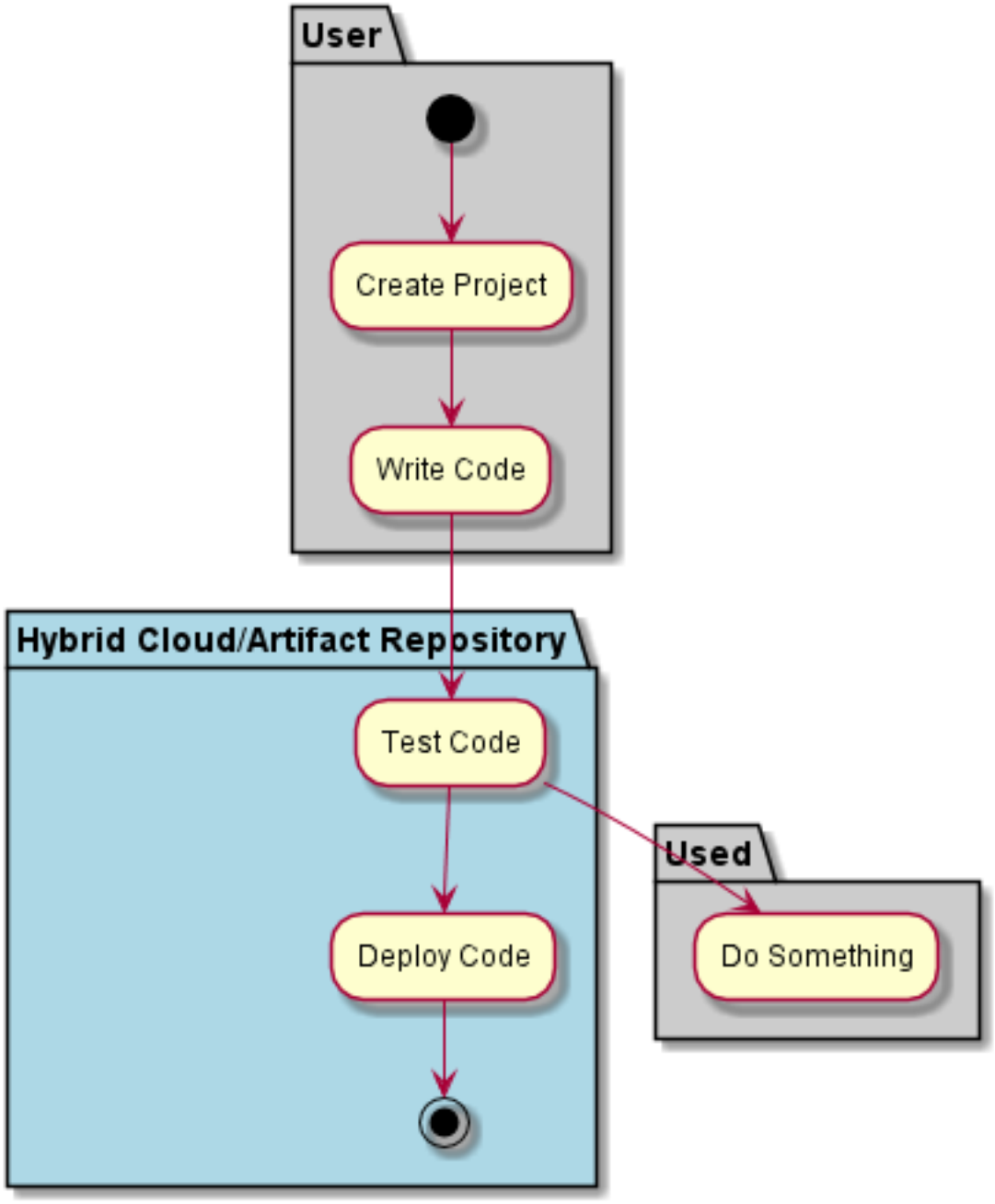
Logical Artifacts

-



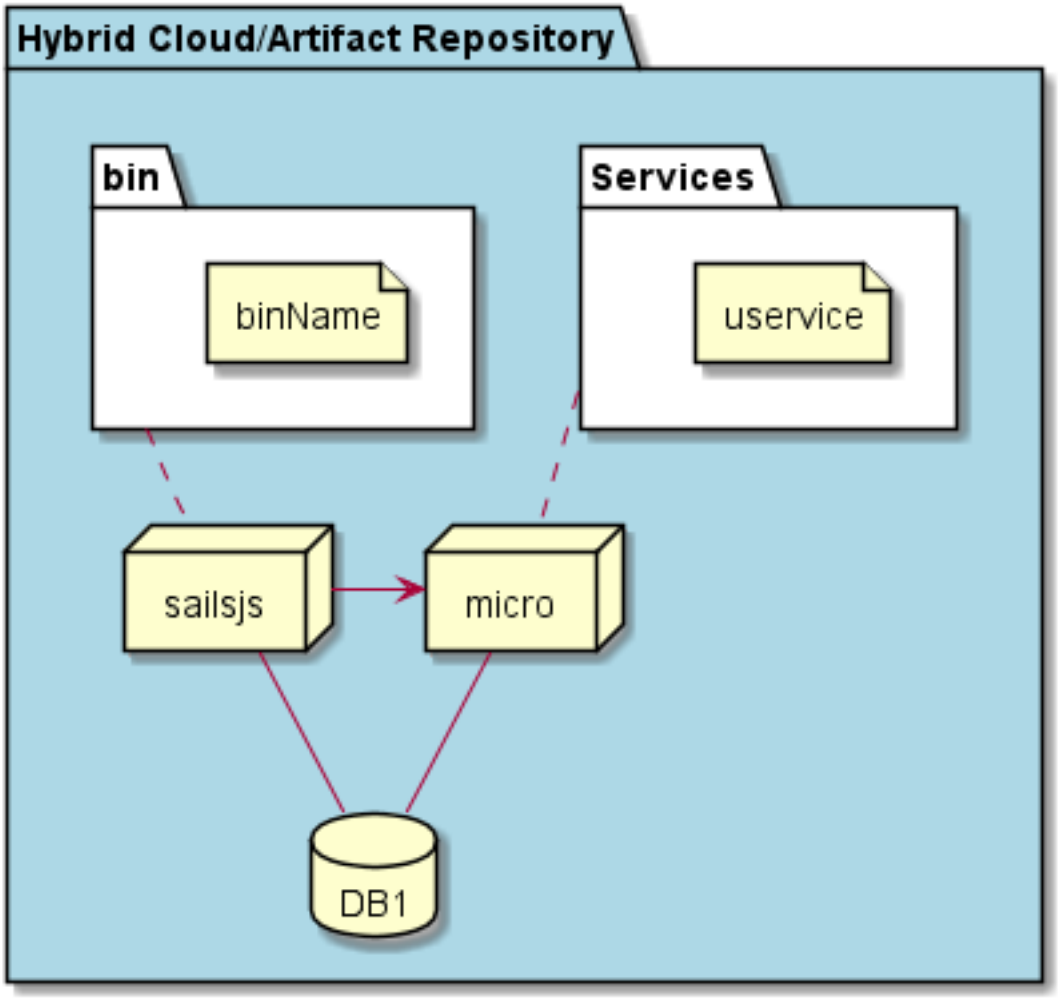
Activities and Flows

The Hybrid Cloud/Artifact Repository subsystem provides the following activities and flows.



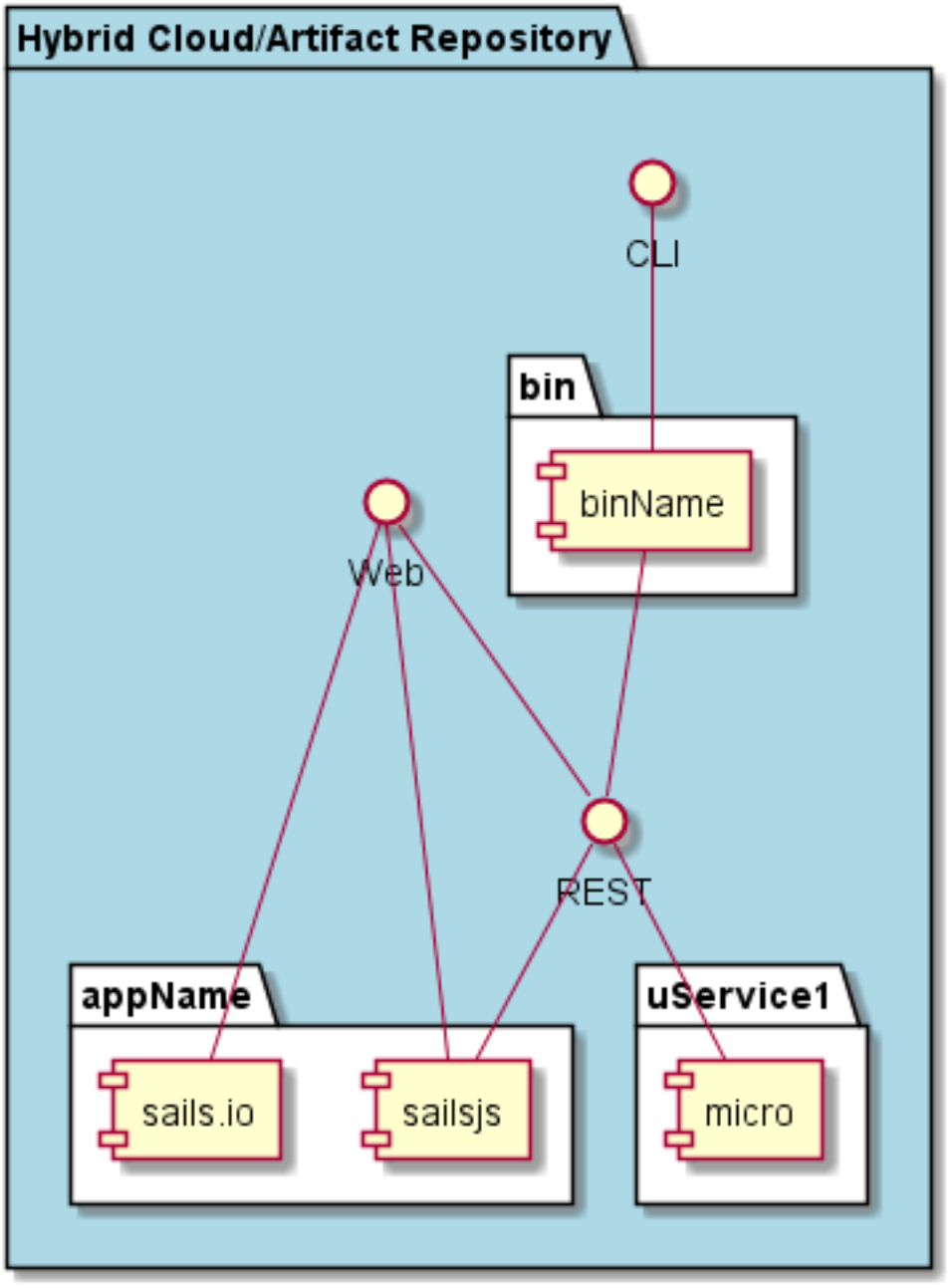
Deployment Architecture

This subsystem is deployed using micro-services as shown in the diagram below. The 'micro' module is used to implement the micro-services in the system. The subsystem also has an CLI, REST and Web Interface exposed through a sailajs application. The sailsjs application will interface with the micro-services and can monitor and drive work-flows through the mesh of micro-services.



Physical Architecture

The Hybrid Cloud/Artifact Repository subsystem is is physically laid out on a hybrid cloud infrastructure. Each microservice is shown how they connect to each other. All of the micro-services communicate to each other and the main app through a REST interface. A CLI, REST or Web interface for the app is how other subsystems or actors interact. Requests are forwarded to micro-services through the REST interface of each micro-service.



Micro-Services

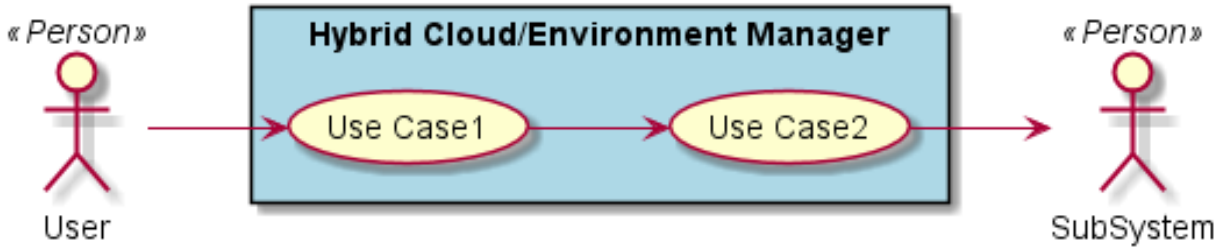
-

2.3.4 Hybrid Cloud/Environment Manager

Hybrid Cloud/Environment Manager is a subsystem of caade . . .

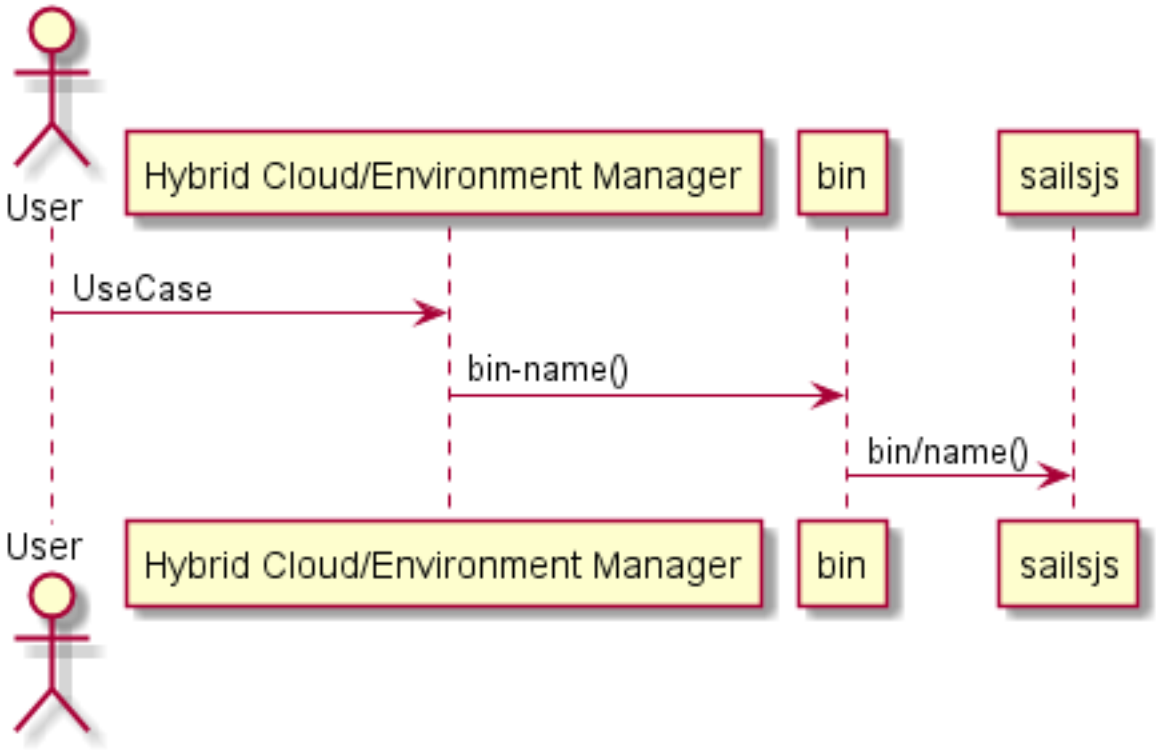
Use Cases

-



Users

- *DevOps*



Uses

- *Hybrid Cloud/Environment Manager*

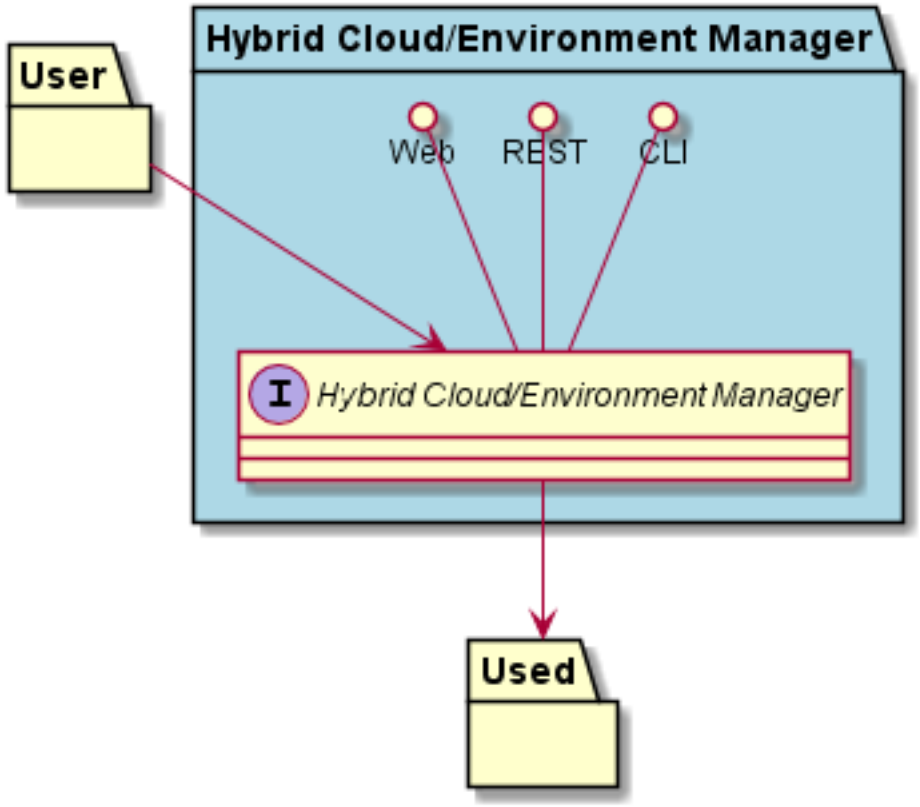
Interface

- CLI - Command Line Interface
- REST-API -

- Portal - Web Portal

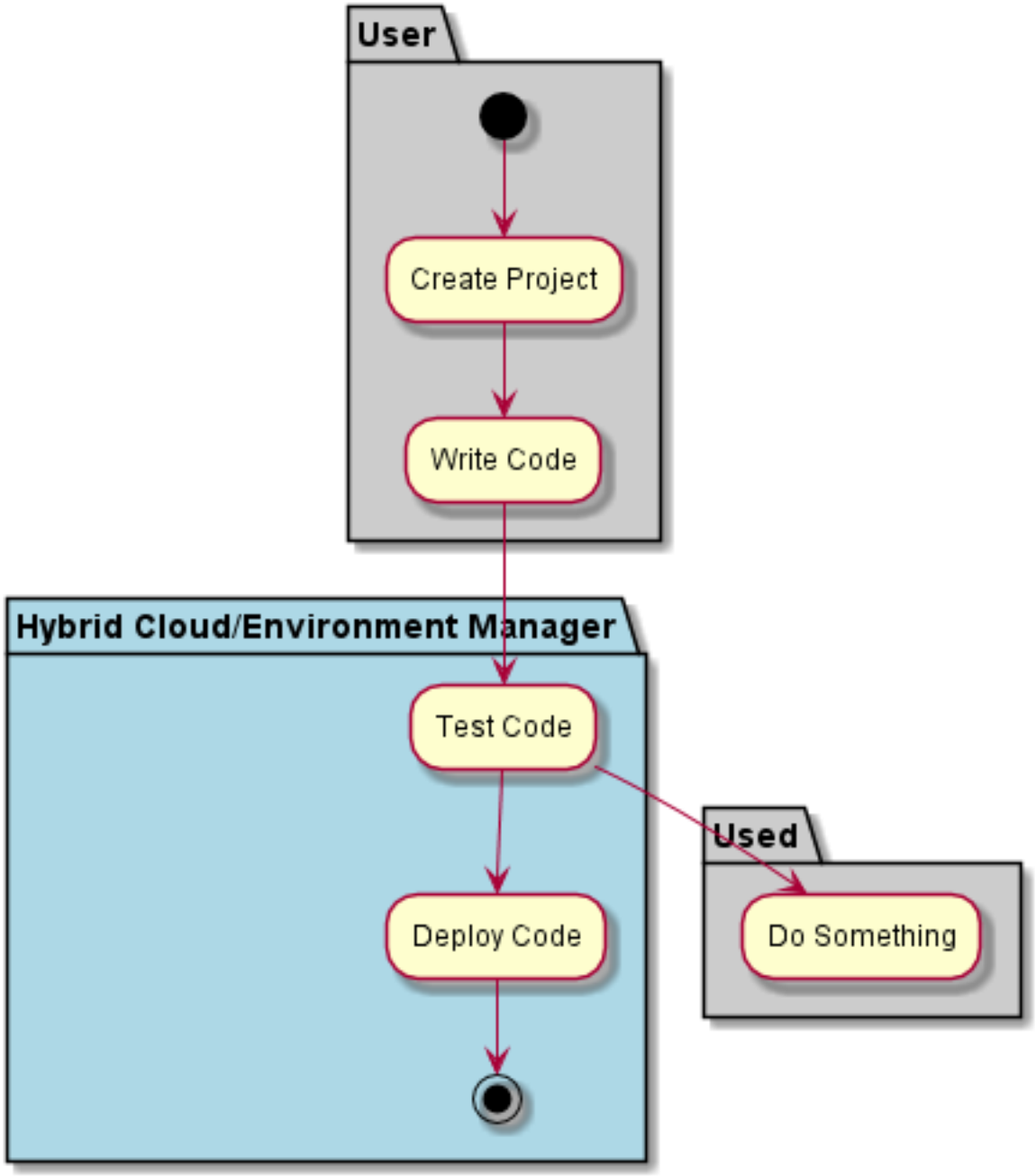
Logical Artifacts

-



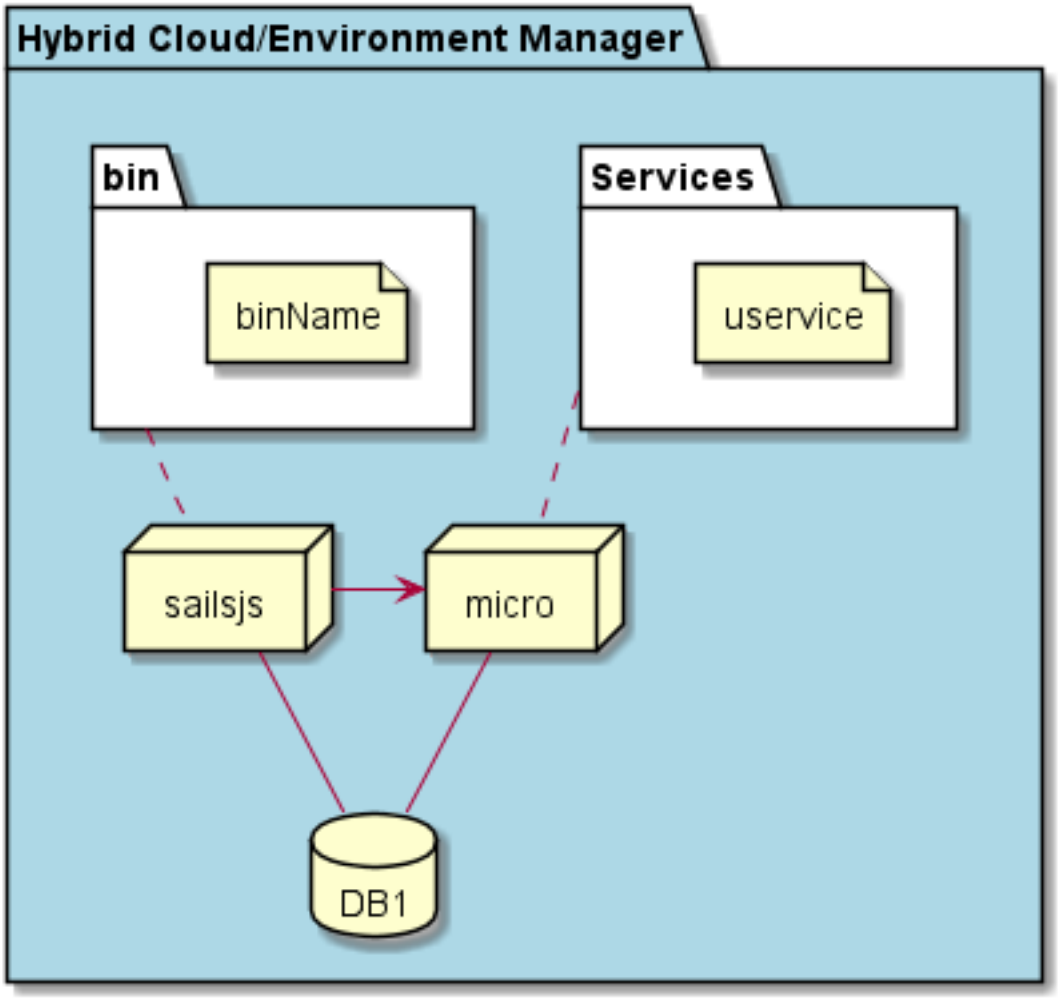
Activities and Flows

The Hybrid Cloud/Environment Manager subsystem provides the following activities and flows.



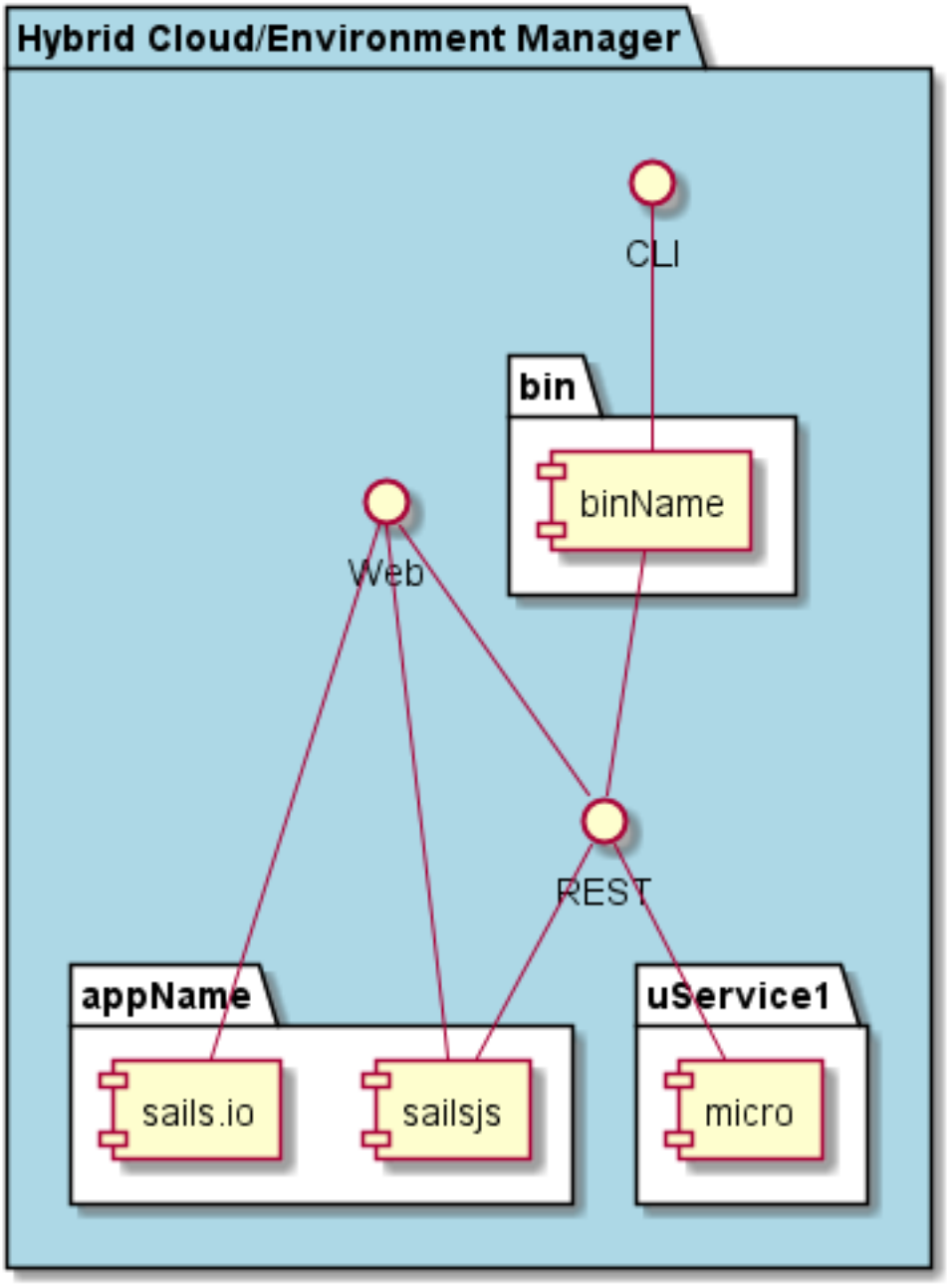
Deployment Architecture

This subsystem is deployed using micro-services as shown in the diagram below. The 'micro' module is used to implement the micro-services in the system. The subsystem also has an CLI, REST and Web Interface exposed through a sailajs application. The sailsjs application will interface with the micro-services and can monitor and drive work-flows through the mesh of micro-services.



Physical Architecture

The Hybrid Cloud/Environment Manager subsystem is is physically laid out on a hybrid cloud infrastructure. Each microservice is shown how they connect to each other. All of the micro-services communicate to each other and the main app through a REST interface. A CLI, REST or Web interface for the app is how other subsystems or actors interact. Requests are forwarded to micro-services through the REST interface of each micro-service.



Micro-Services

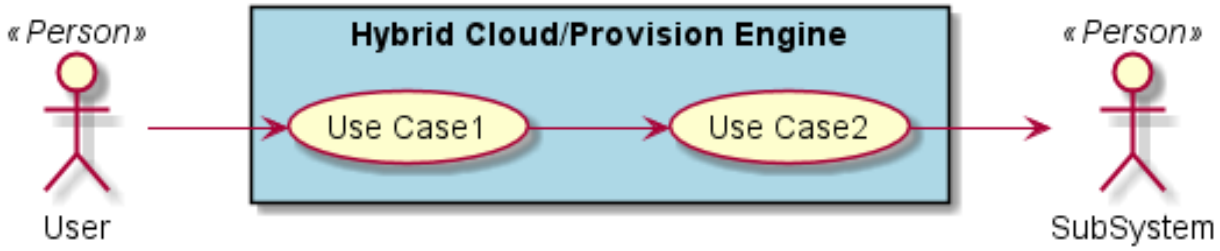
-

2.3.5 Hybrid Cloud/Provision Engine

Hybrid Cloud/Provision Engine is a subsystem of caade . . .

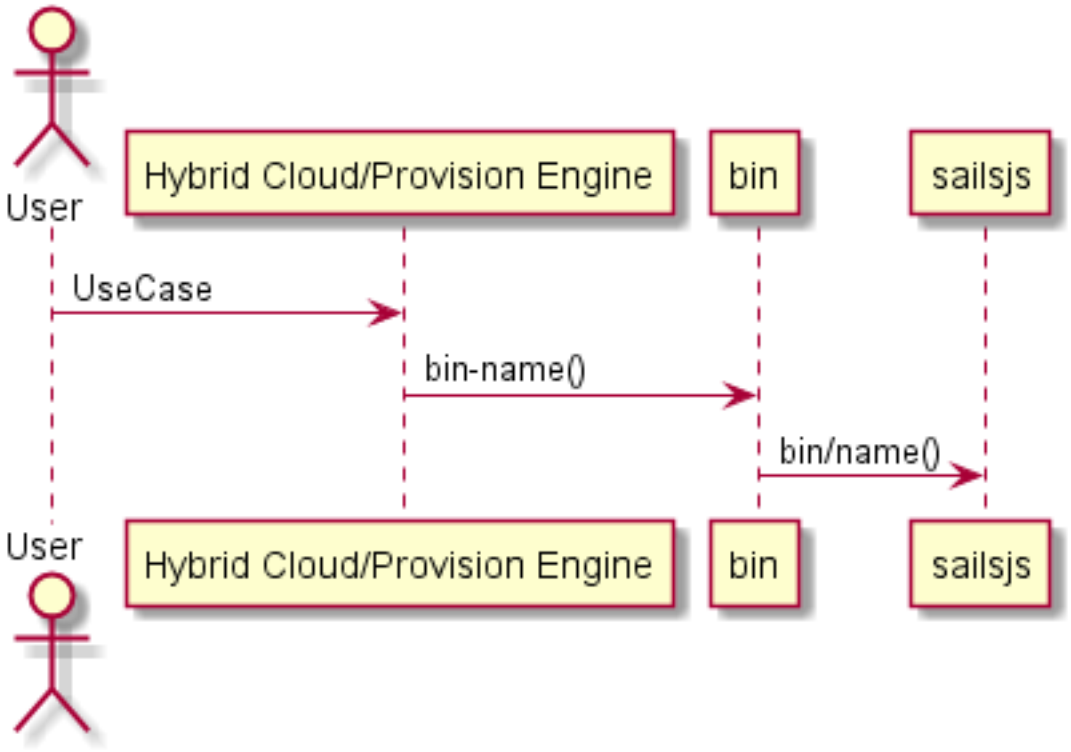
Use Cases

-



Users

- *DevOps*



Uses

- *Hybrid Cloud/Provision Engine*

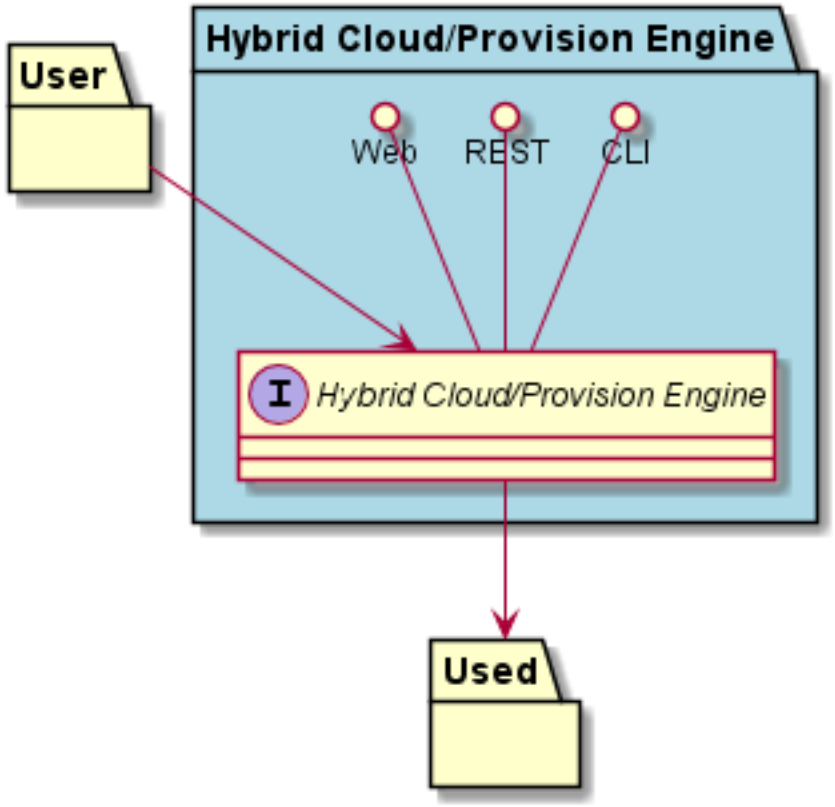
Interface

- CLI - Command Line Interface
- REST-API -

- Portal - Web Portal

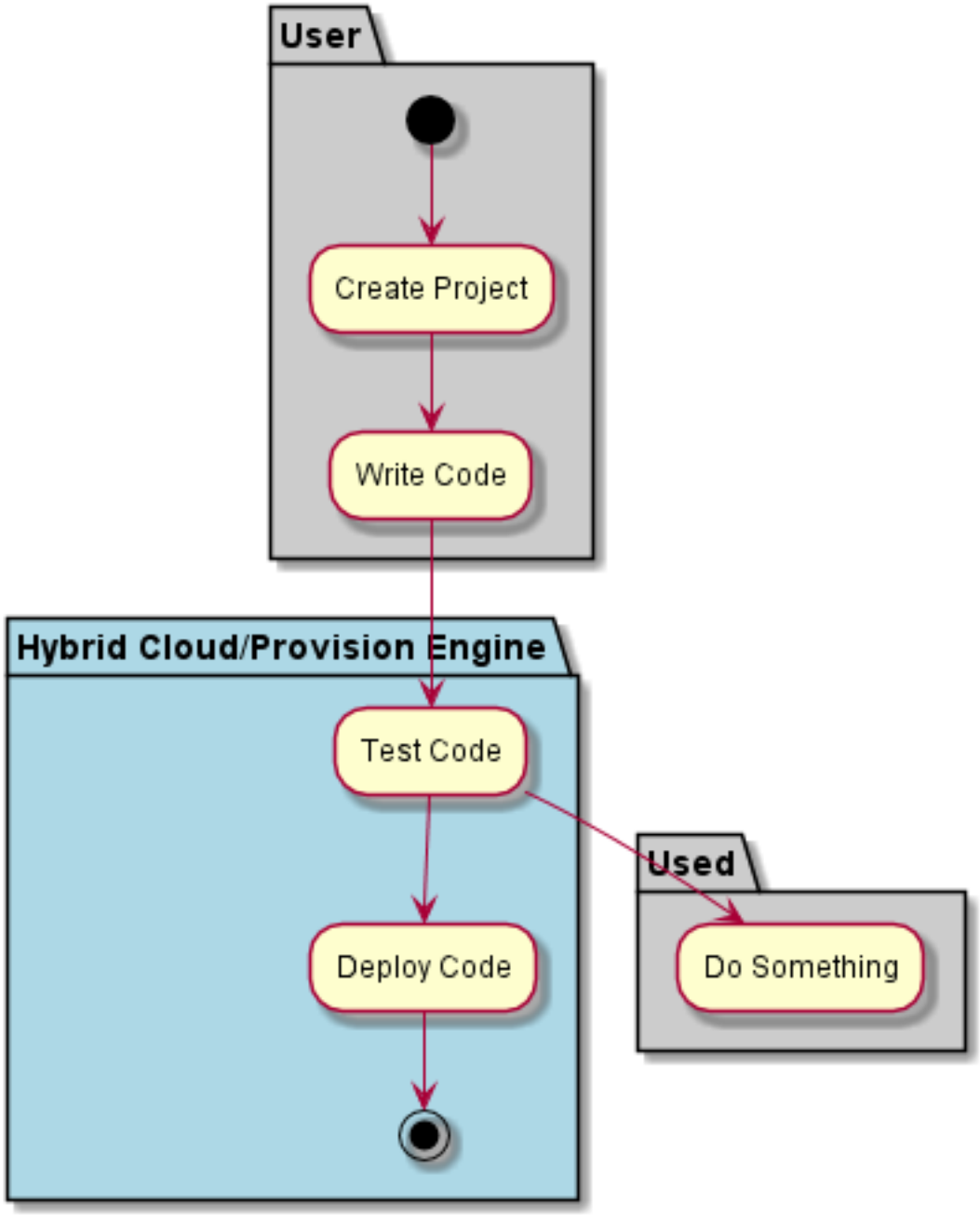
Logical Artifacts

-



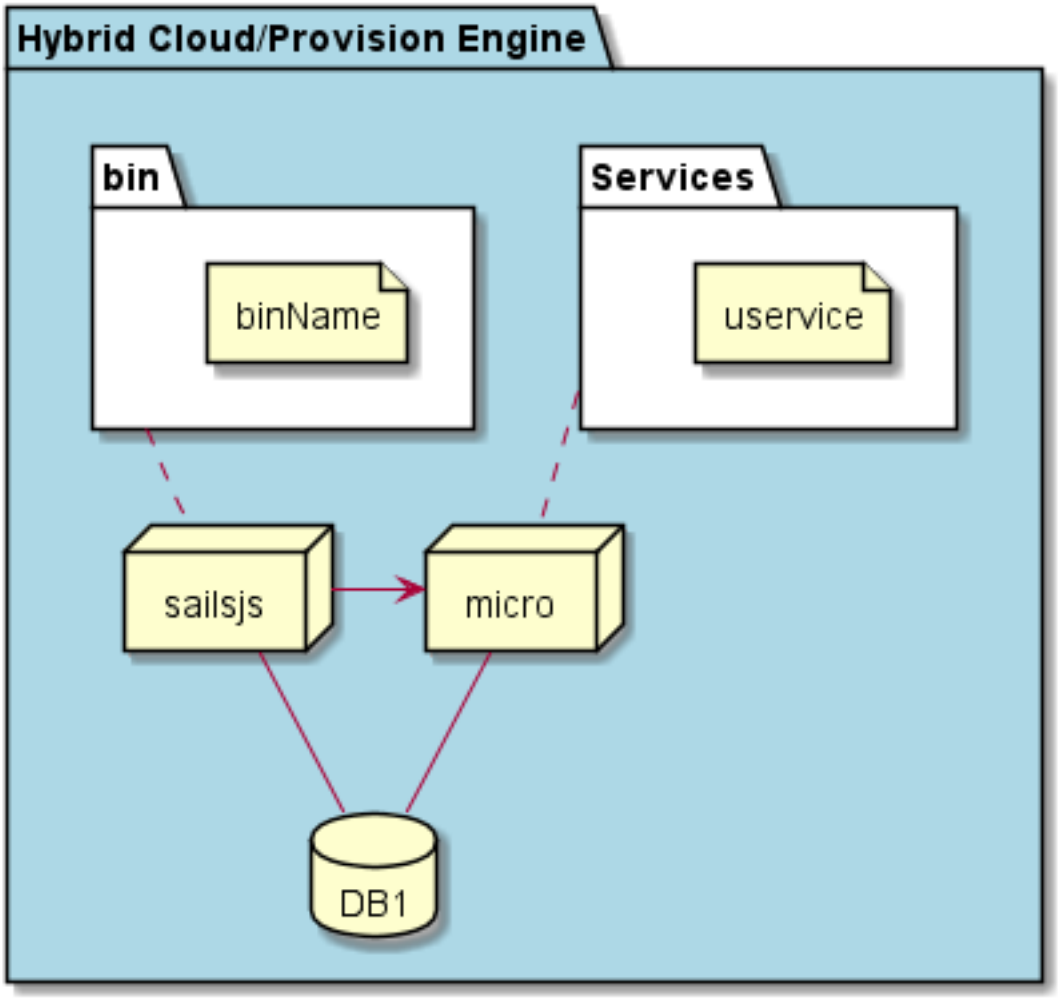
Activities and Flows

The Hybrid Cloud/Provision Engine subsystem provides the following activities and flows.



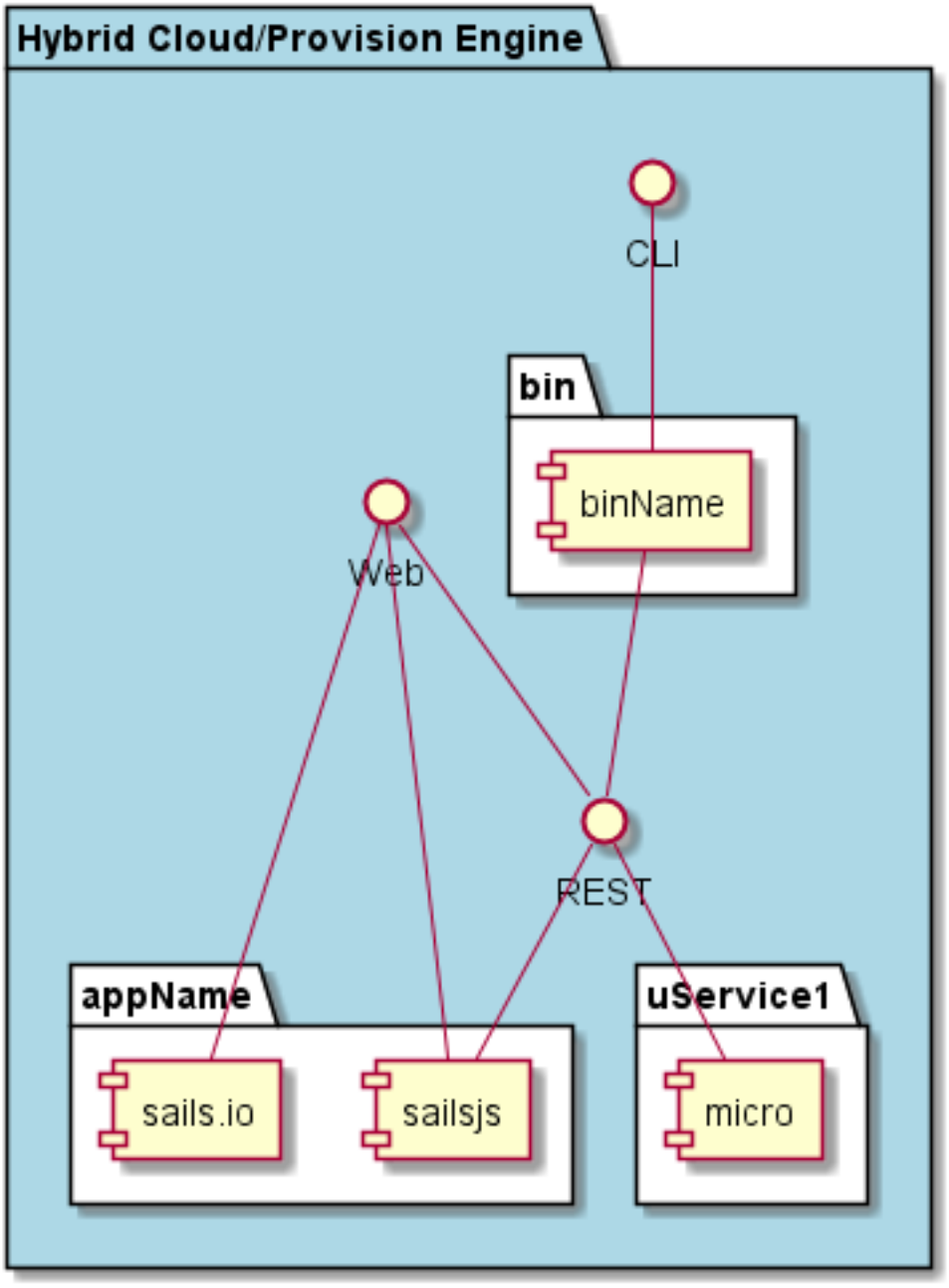
Deployment Architecture

This subsystem is deployed using micro-services as shown in the diagram below. The 'micro' module is used to implement the micro-services in the system. The subsystem also has an CLI, REST and Web Interface exposed through a sailajs application. The sailsjs application will interface with the micro-services and can monitor and drive work-flows through the mesh of micro-services.



Physical Architecture

The Hybrid Cloud/Provision Engine subsystem is is physically laid out on a hybrid cloud infrastructure. Each microservice is shown how they connect to each other. All of the micro-services communicate to each other and the main app through a REST interface. A CLI, REST or Web interface for the app is how other subsystems or actors interact. Requests are forwarded to micro-services through the REST interface of each micro-service.



Micro-Services

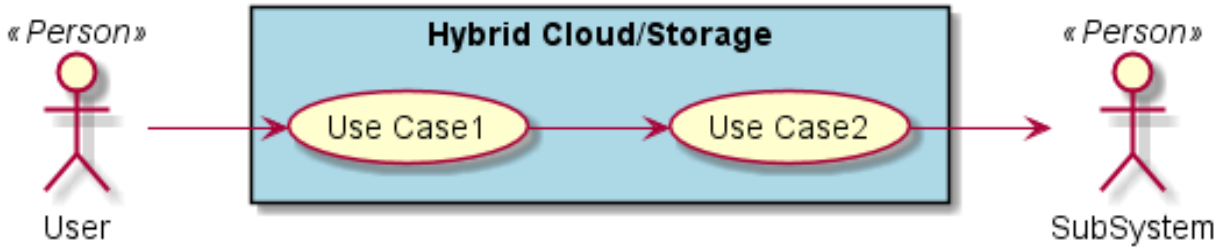
-

2.3.6 Hybrid Cloud/Storage

Hybrid Cloud/Storage is a subsystem of caade …

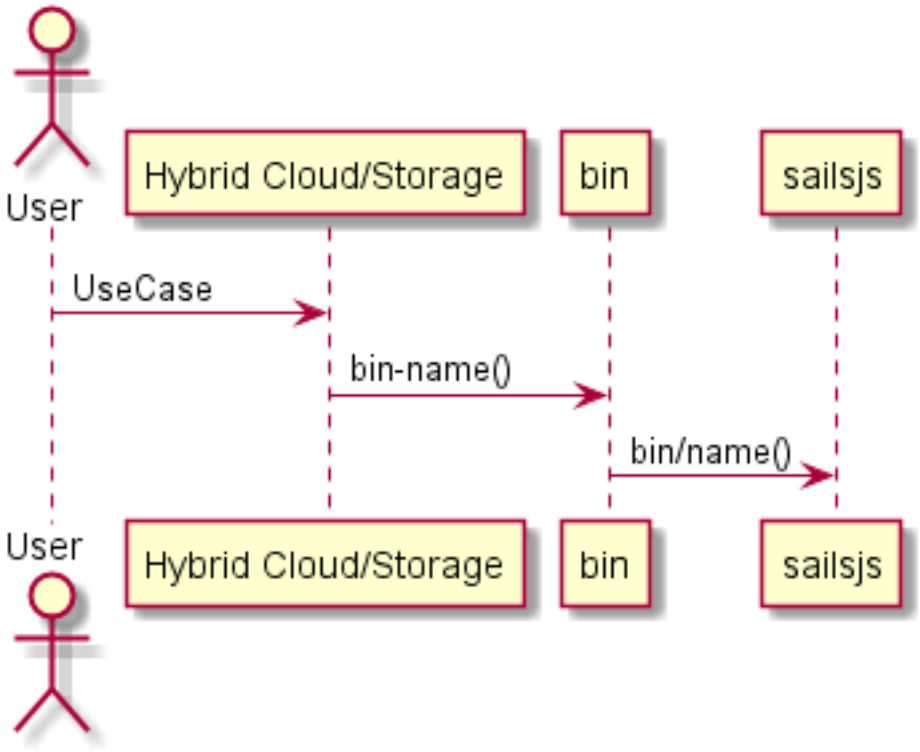
Use Cases

-



Users

- *DevOps*



Uses

- *Hybrid Cloud/Storage*

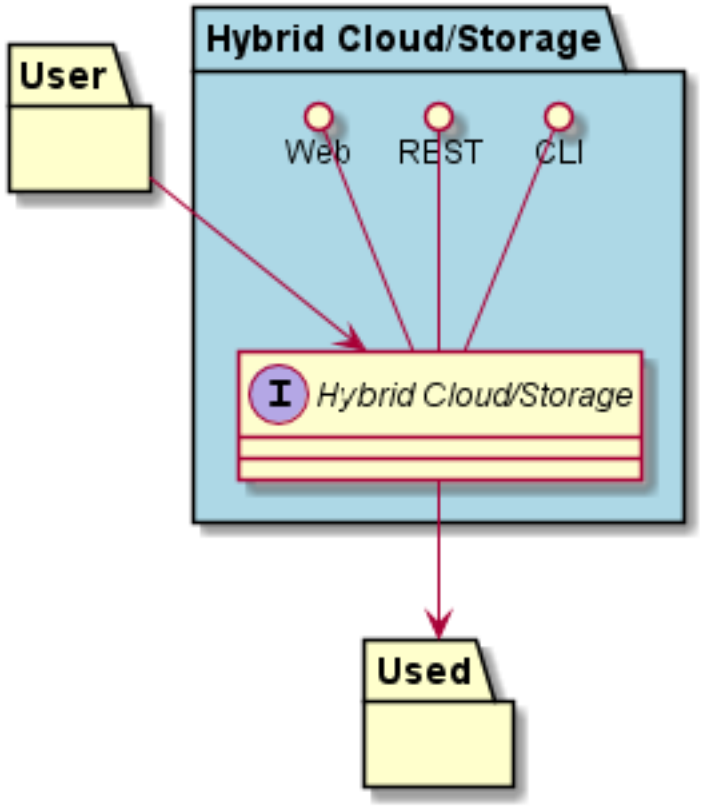
Interface

- CLI - Command Line Interface
- REST-API -

- Portal - Web Portal

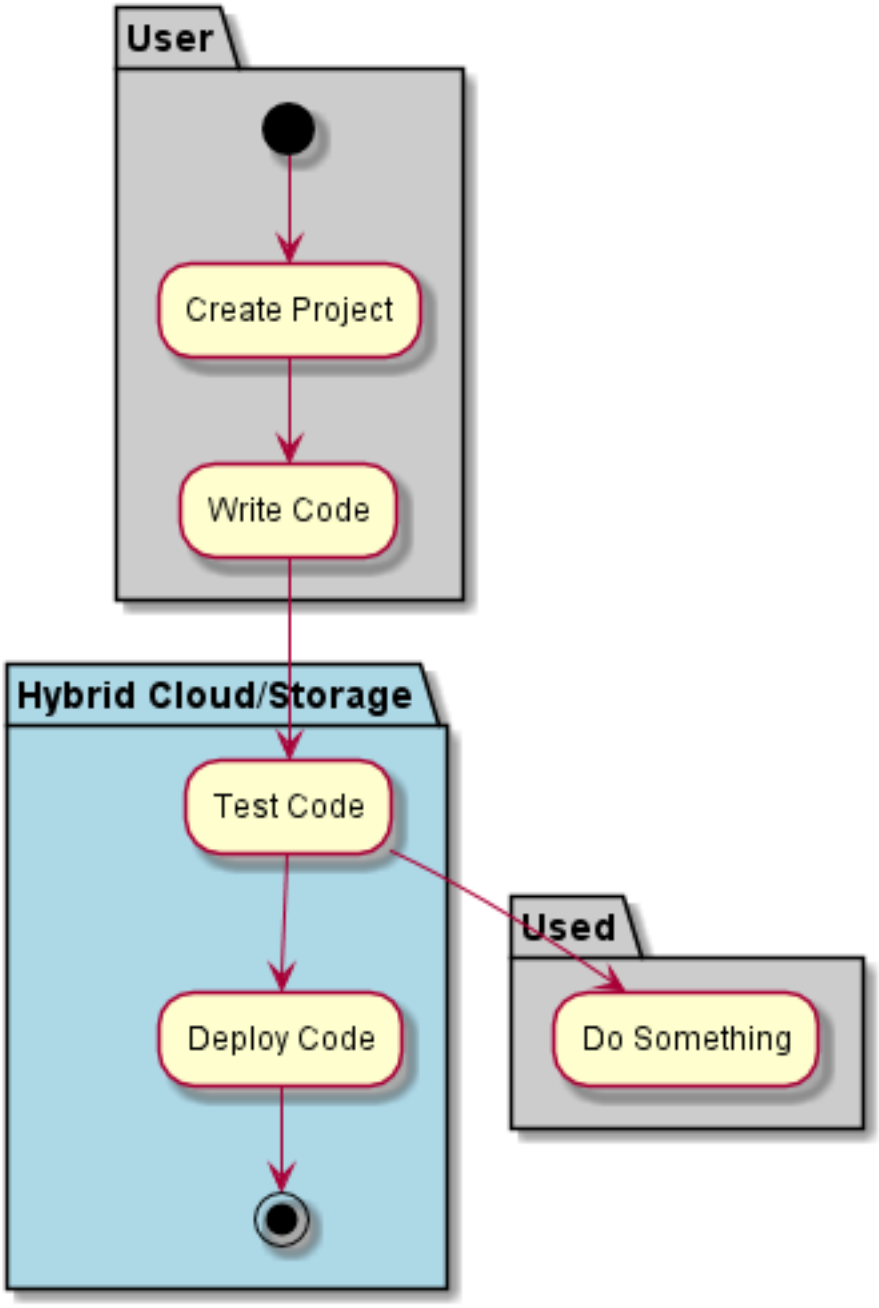
Logical Artifacts

-



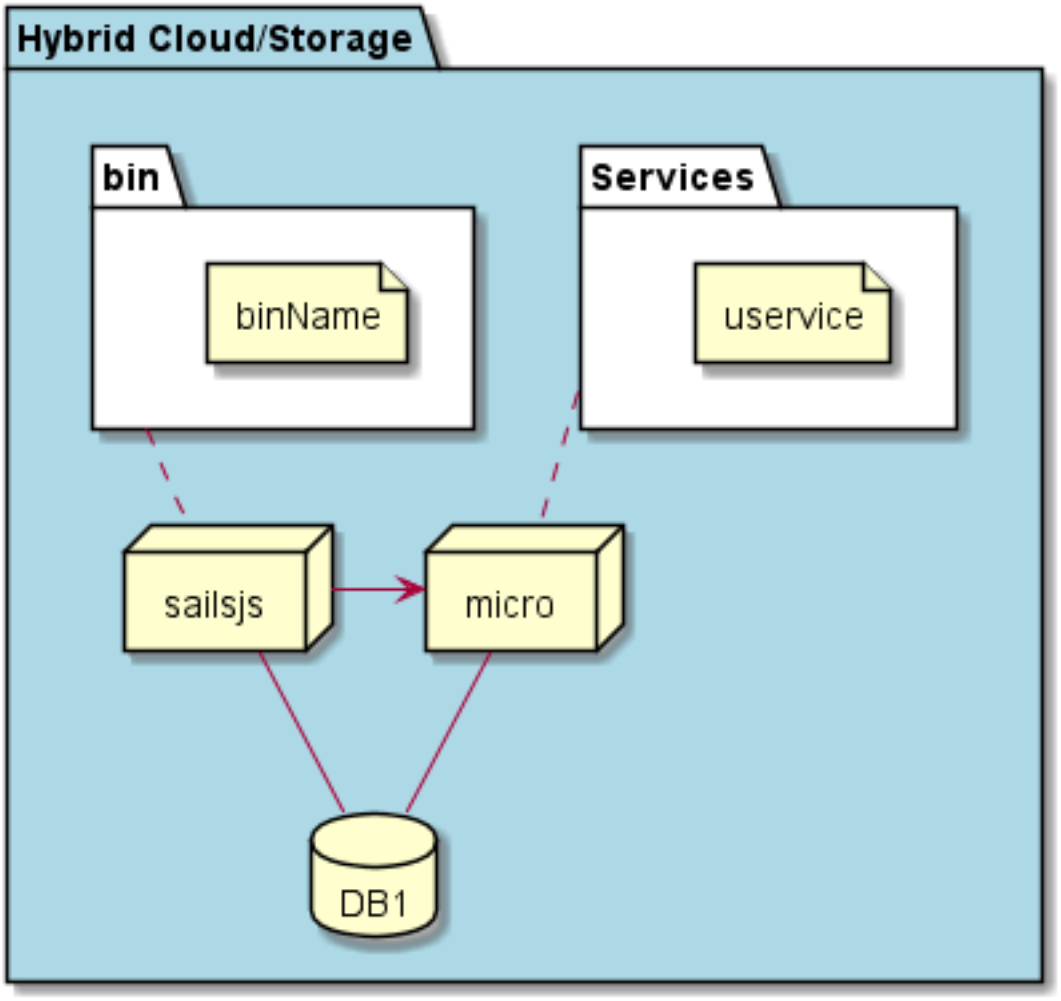
Activities and Flows

The Hybrid Cloud/Storage subsystem provides the following activities and flows.



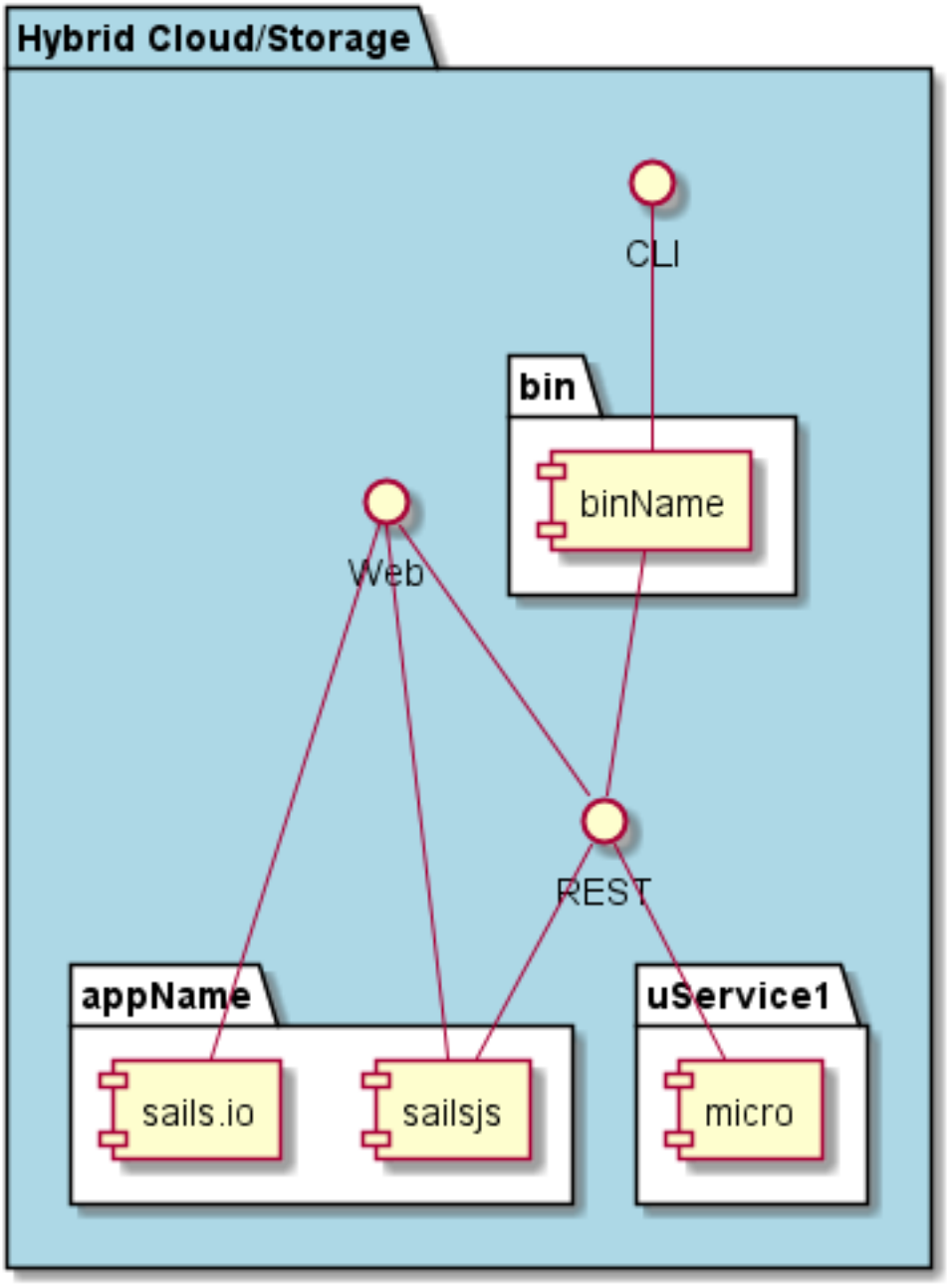
Deployment Architecture

This subsystem is deployed using micro-services as shown in the diagram below. The 'micro' module is used to implement the micro-services in the system. The subsystem also has an CLI, REST and Web Interface exposed through a sailajs application. The sailsjs application will interface with the micro-services and can monitor and drive work-flows through the mesh of micro-services.



Physical Architecture

The Hybrid Cloud/Storage subsystem is is physically laid out on a hybrid cloud infrastructure. Each microservice is shown how they connect to each other. All of the micro-services communicate to each other and the main app through a REST interface. A CLI, REST or Web interface for the app is how other subsystems or actors interact. Requests are forwarded to micro-services through the REST interface of each micro-service.



Micro-Services

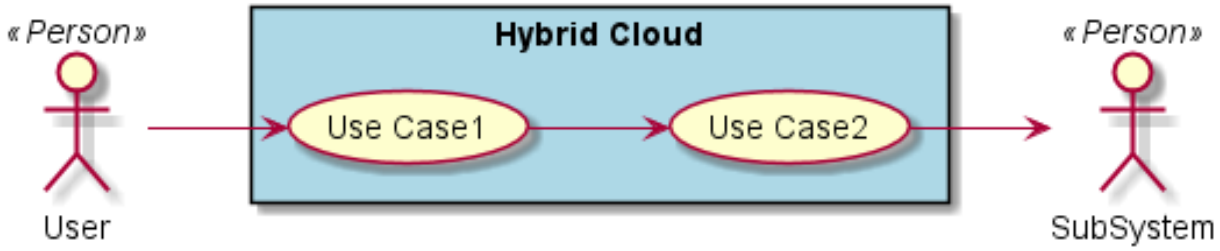
-

2.3.7 Hybrid Cloud

Hybrid Cloud is a subsystem of caade . . .

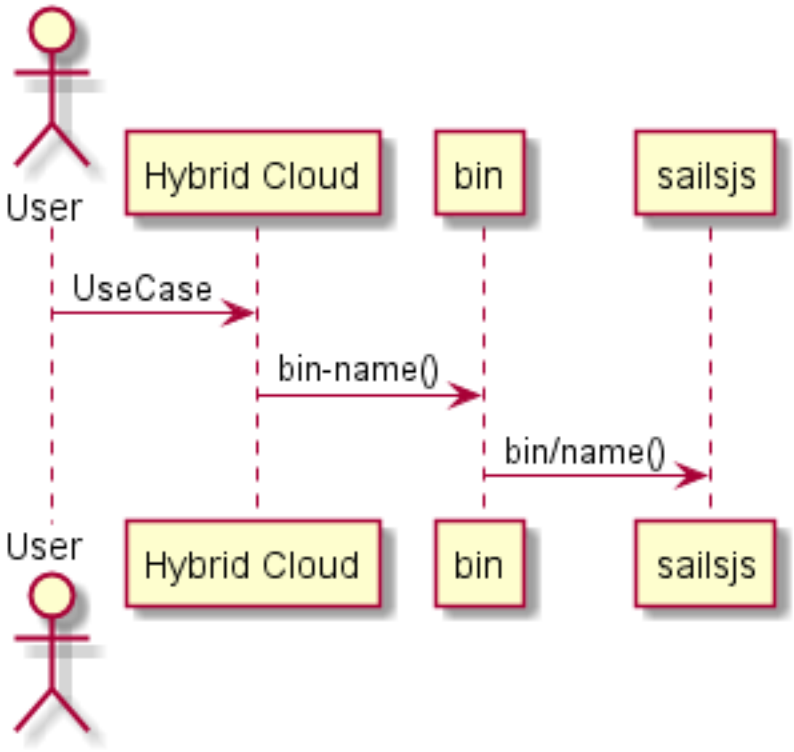
Use Cases

-



Users

- *DevOps*



Uses

- *Hybrid Cloud*

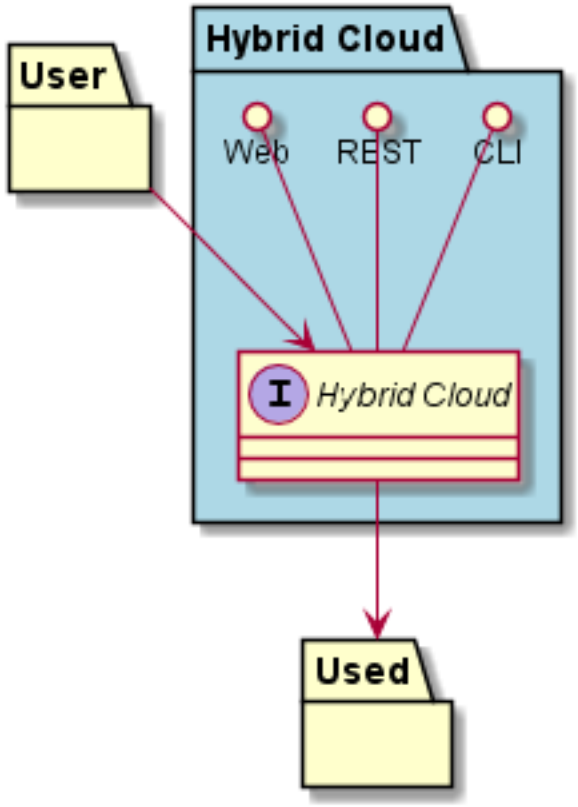
Interface

- CLI - Command Line Interface
- REST-API -

- Portal - Web Portal

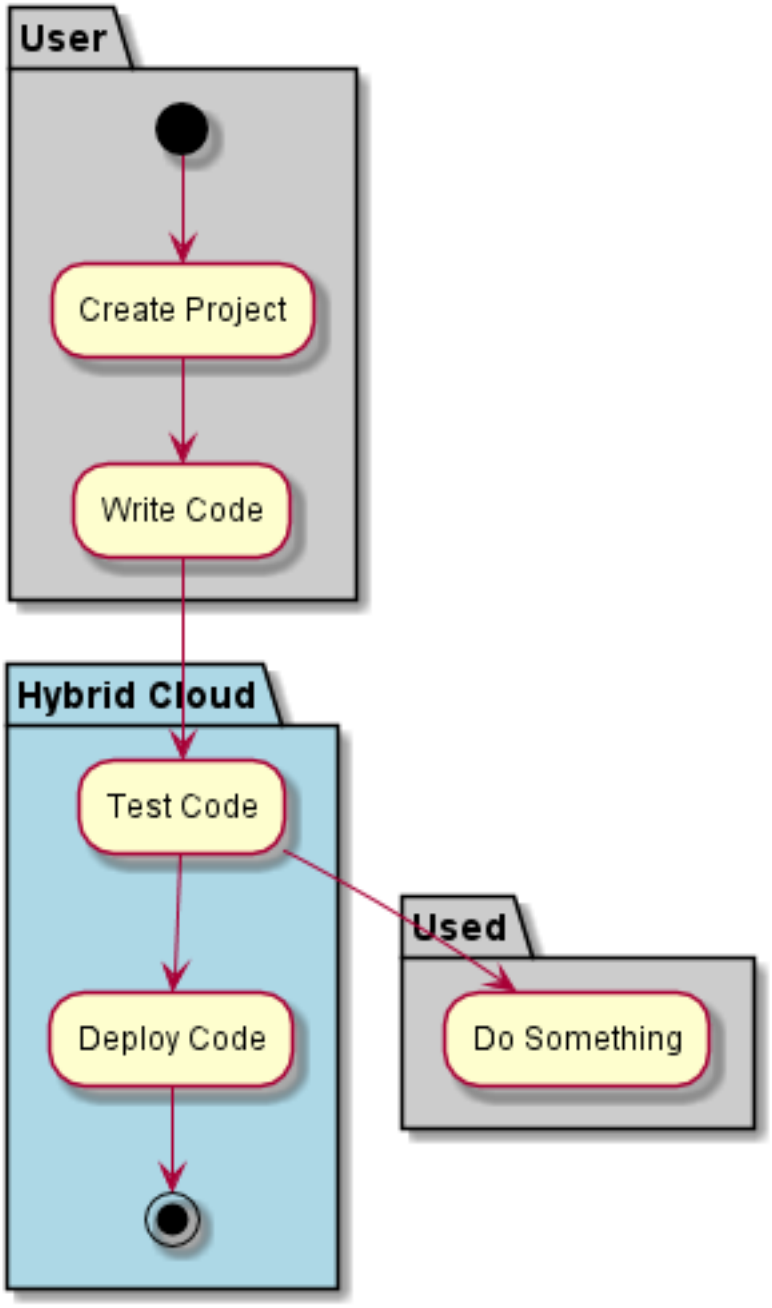
Logical Artifacts

-



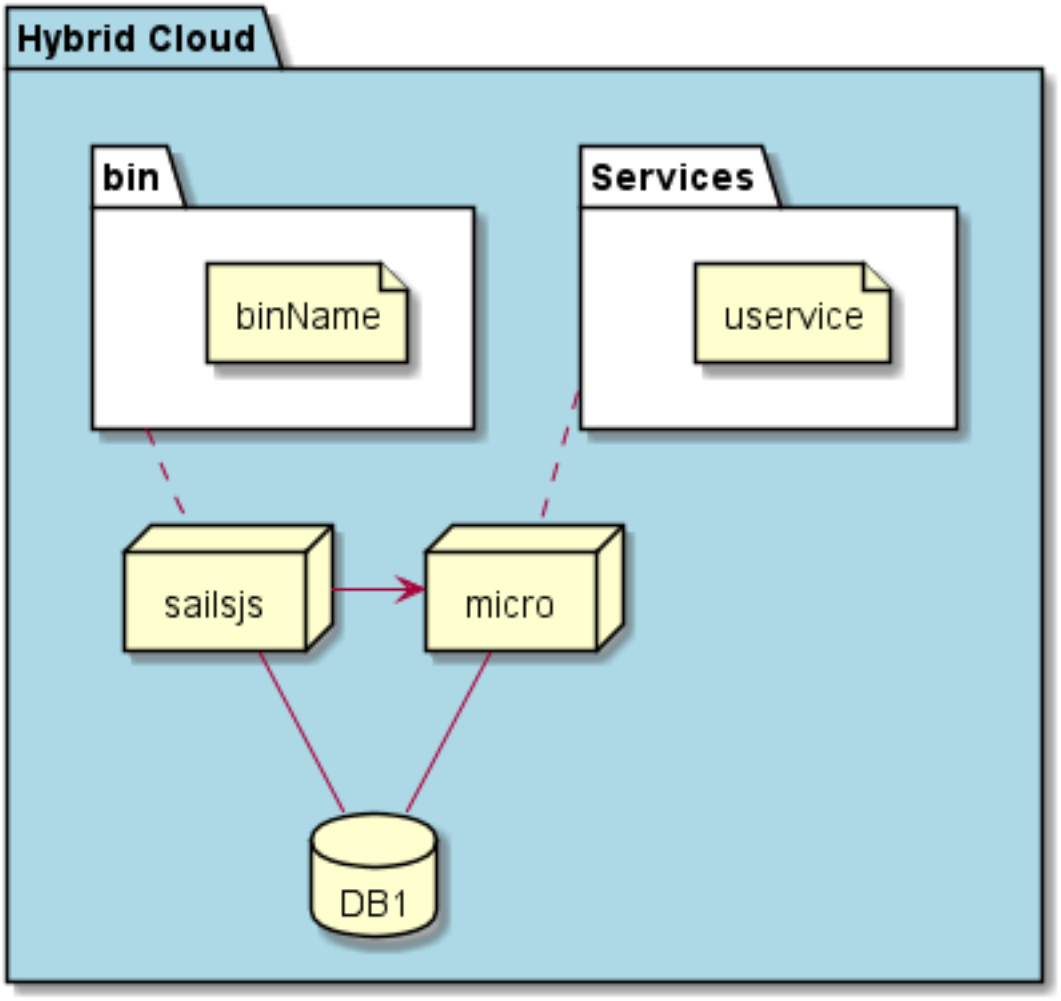
Activities and Flows

The Hybrid Cloud subsystem provides the following activities and flows.



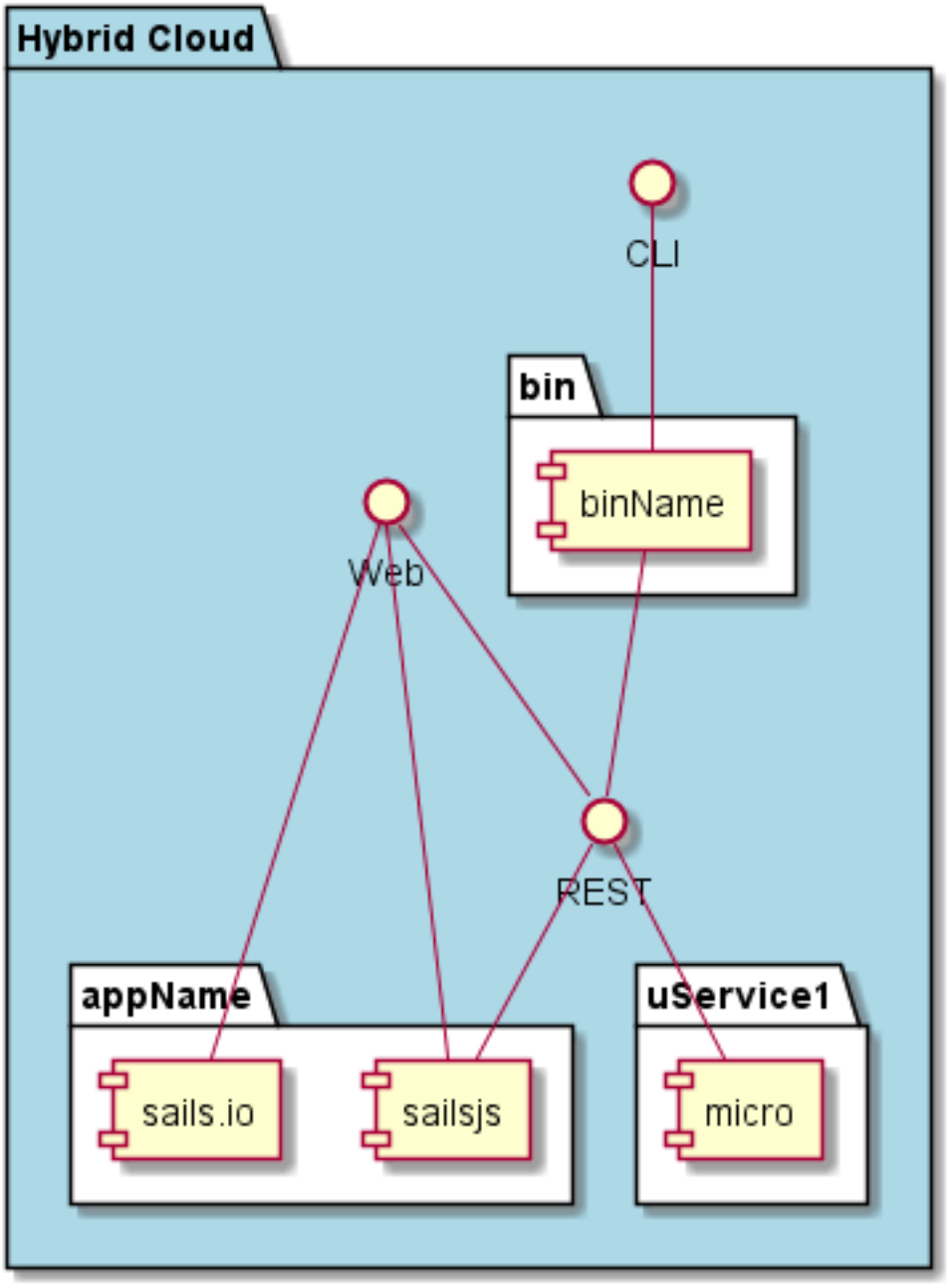
Deployment Architecture

This subsystem is deployed using micro-services as shown in the diagram below. The 'micro' module is used to implement the micro-services in the system. The subsystem also has an CLI, REST and Web Interface exposed through a sailajs application. The sailsjs application will interface with the micro-services and can monitor and drive work-flows through the mesh of micro-services.



Physical Architecture

The Hybrid Cloud subsystem is is physically laid out on a hybrid cloud infrastructure. Each microservice is shown how they connect to each other. All of the micro-services communicate to each other and the main app through a REST interface. A CLI, REST or Web interface for the app is how other subsystems or actors interact. Requests are forwarded to micro-services through the REST interface of each micro-service.



Micro-Services

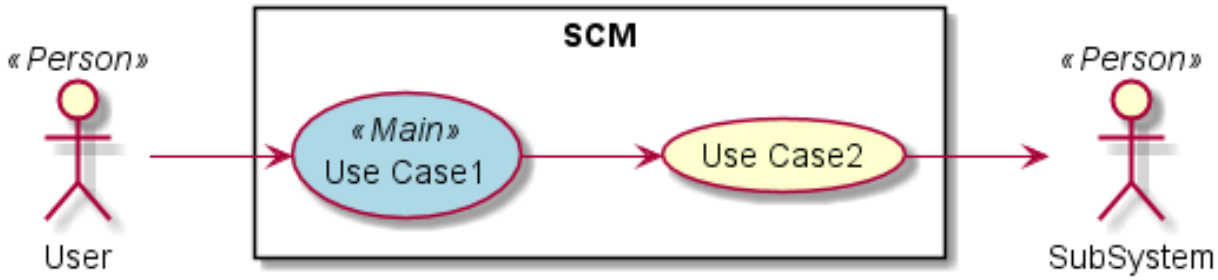
-

2.3.8 SCM

SCM is a subsystem of caade . . .

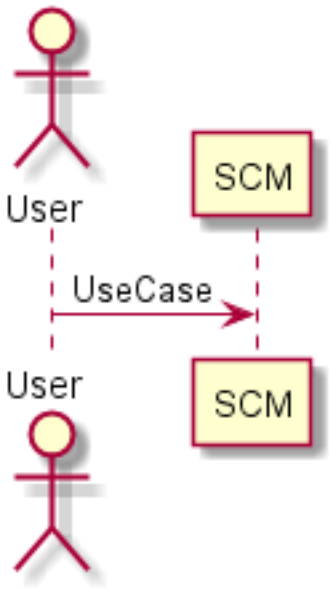
Use Cases

-



Users

-



Uses

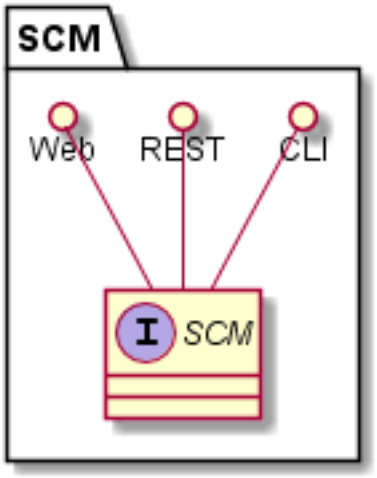
- *Hybrid Cloud*
-

Interface

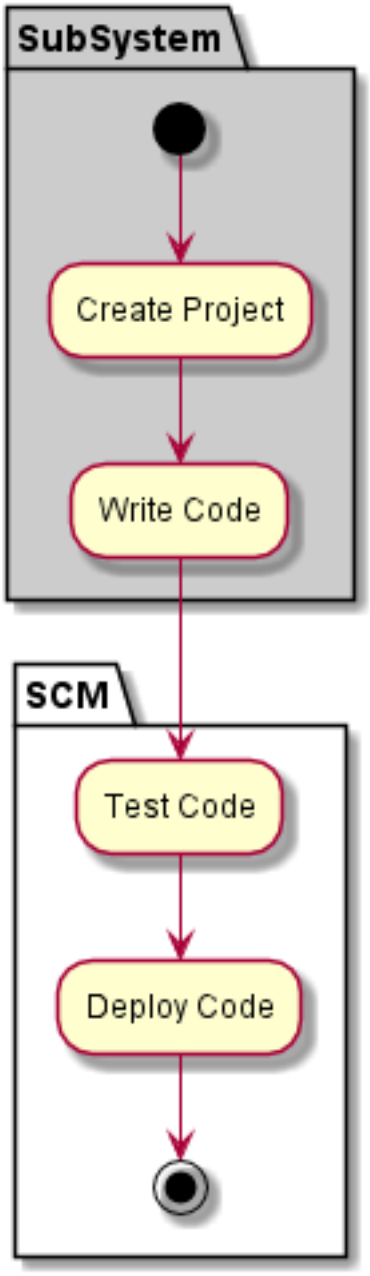
- CLI - Command Line Interface
- REST-API -
- Portal - Web Portal

Logical Artifacts

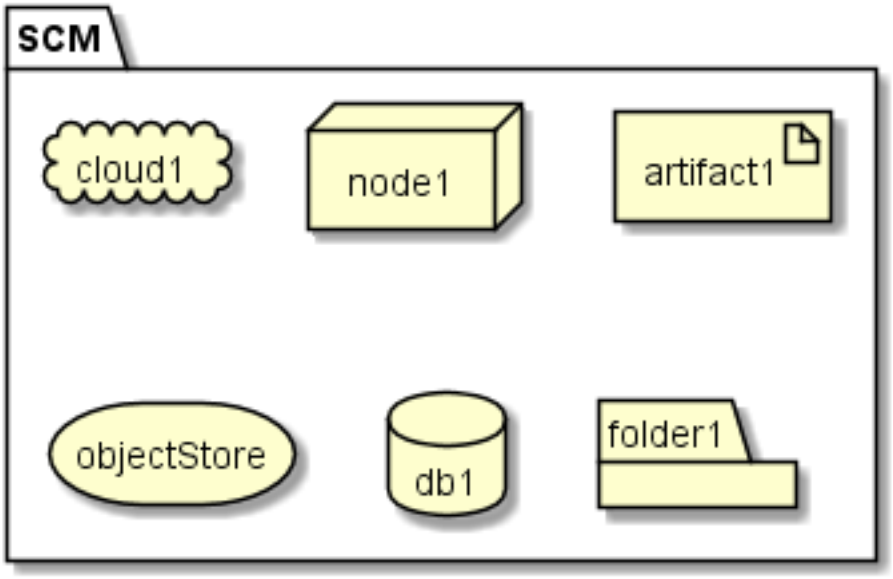
-



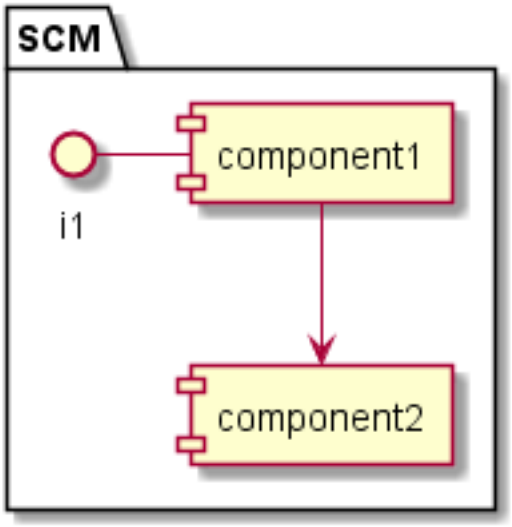
Activities and Flows



Deployment Architecture



Physical Architecture



CHAPTER 3

Actors

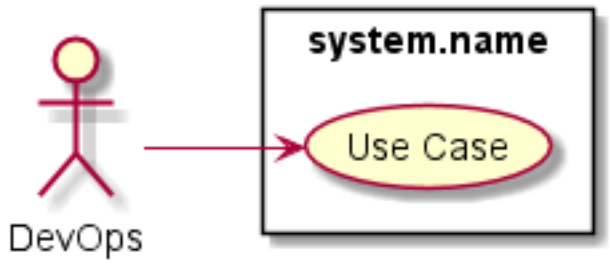
These are the Actors of the CAADE System.

3.1 DevOps

Description

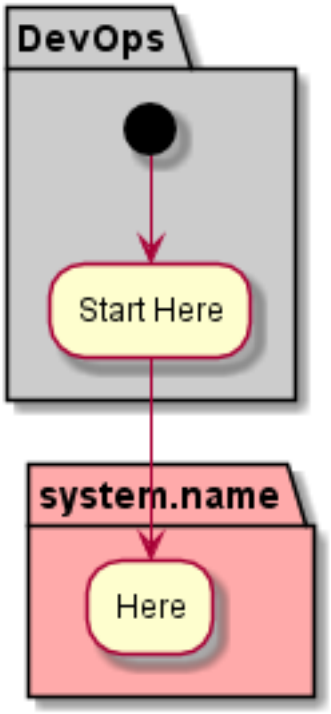
3.1.1 Use Cases

- _UseCases_



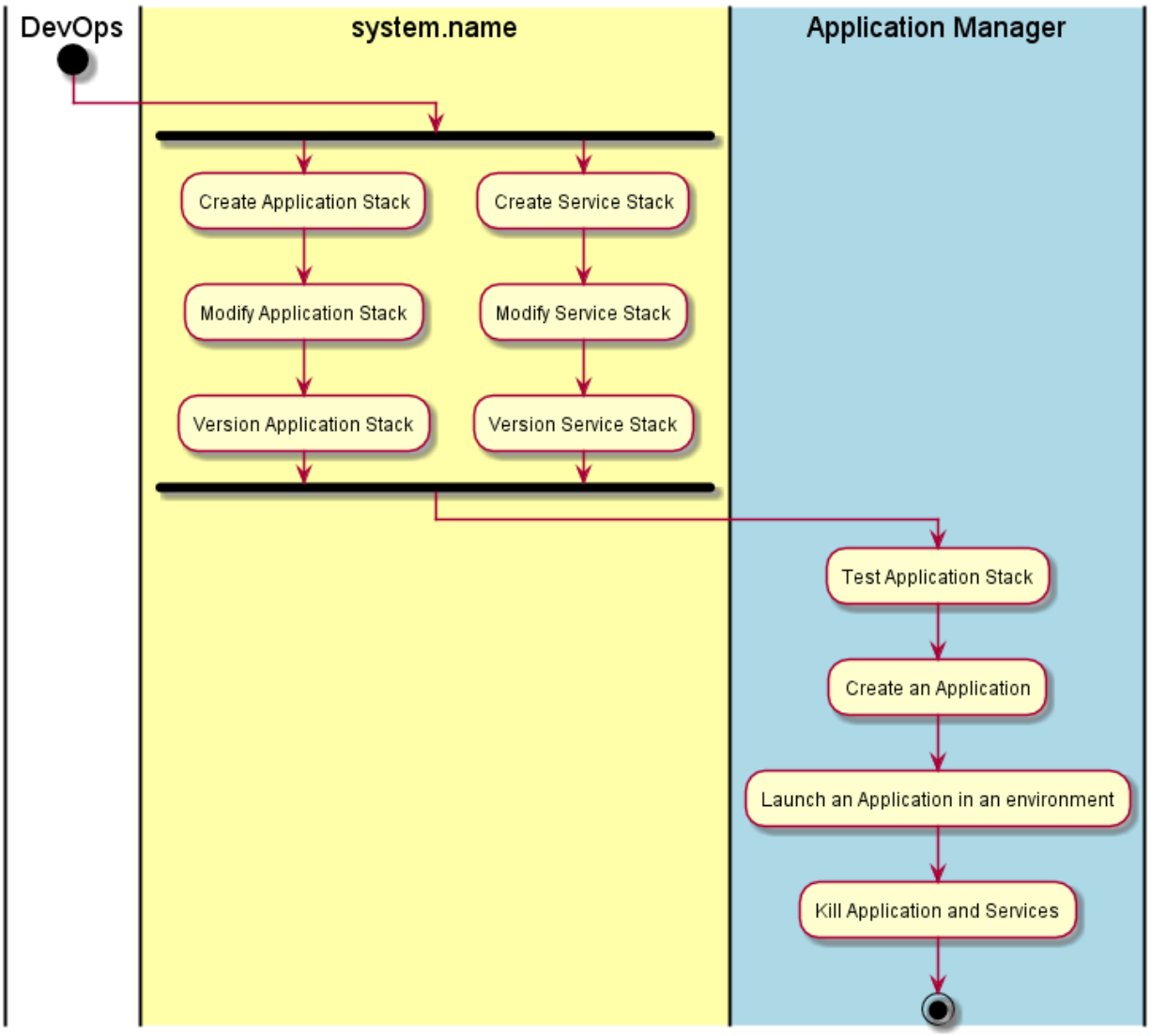
3.1.2 Activities

description



3.1.3 Workflow

description



3.1.4 User Interface

TBD

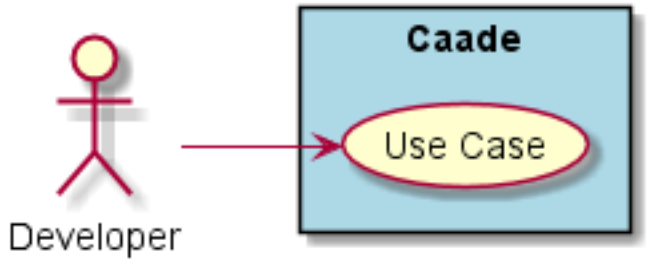
3.1.5 Command Line Interface

TBD

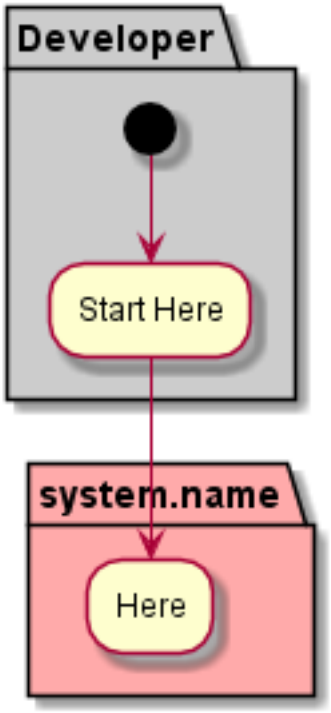
3.2 Developer

DevOps is another actor. Next, *Developer* is another actor.

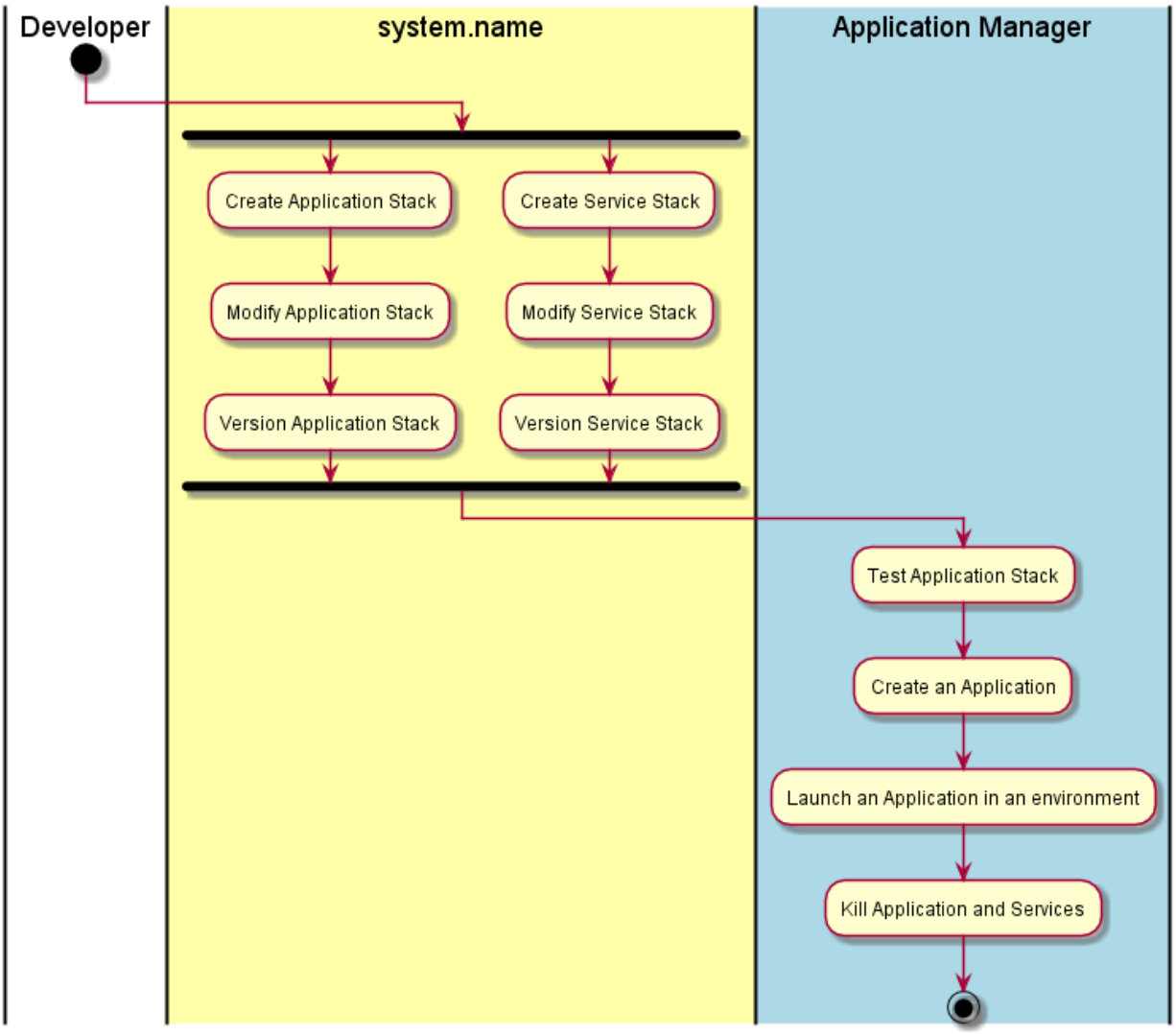
3.2.1 Use Cases



3.2.2 Activities



3.2.3 Workflow



3.2.4 User Interface

TBD

3.2.5 Command Line Interface

TBD

CHAPTER 4

Use Cases

These are the high level usecases for the CAADE Solution

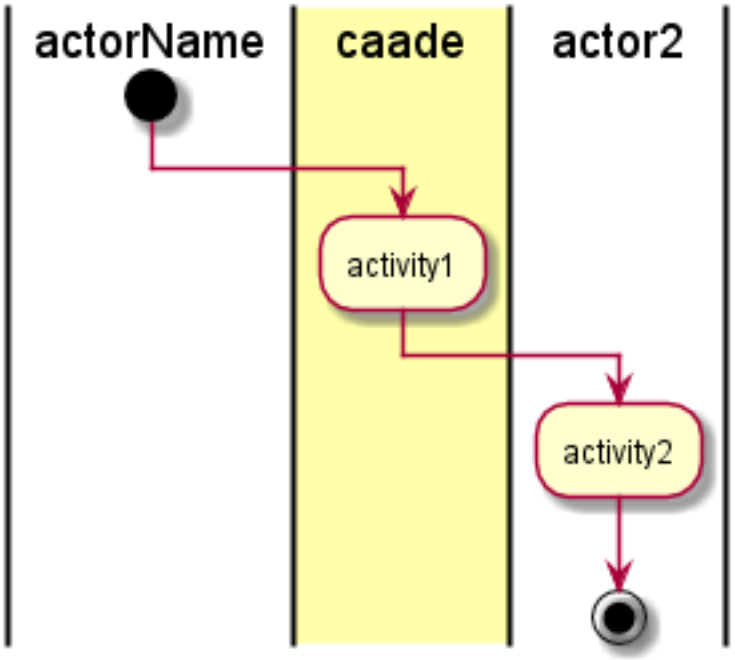
4.1 Manage Application

Add Description

4.1.1 Actors

- *DevOps*

4.1.2 Activities



- Activity from the diagram

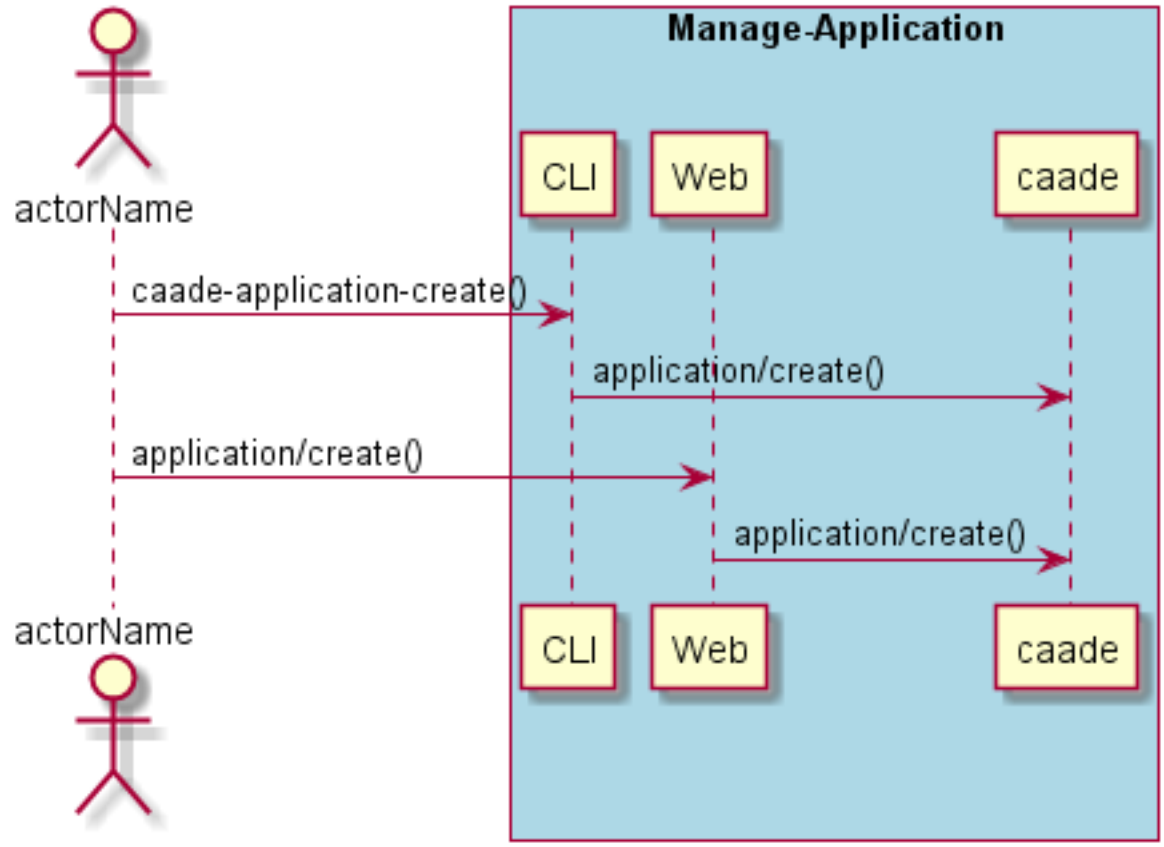
4.1.3 Systems Involved

- *Hybrid Cloud*

4.1.4 Detail Scenarios

Create Application

Create Application using CLI and Web Interface with . . . <parameters>



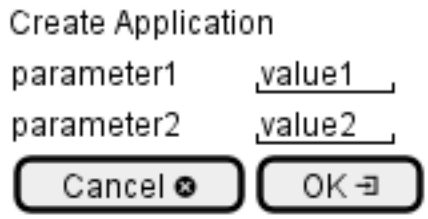
CLI

This is the command line interface for the Create Application Scenario.

```
# caade application create <parameters>
# caade application create exmaple
```

Web Interface

This is a mock up of the Web Interface for the Create Application Scenario.



REST

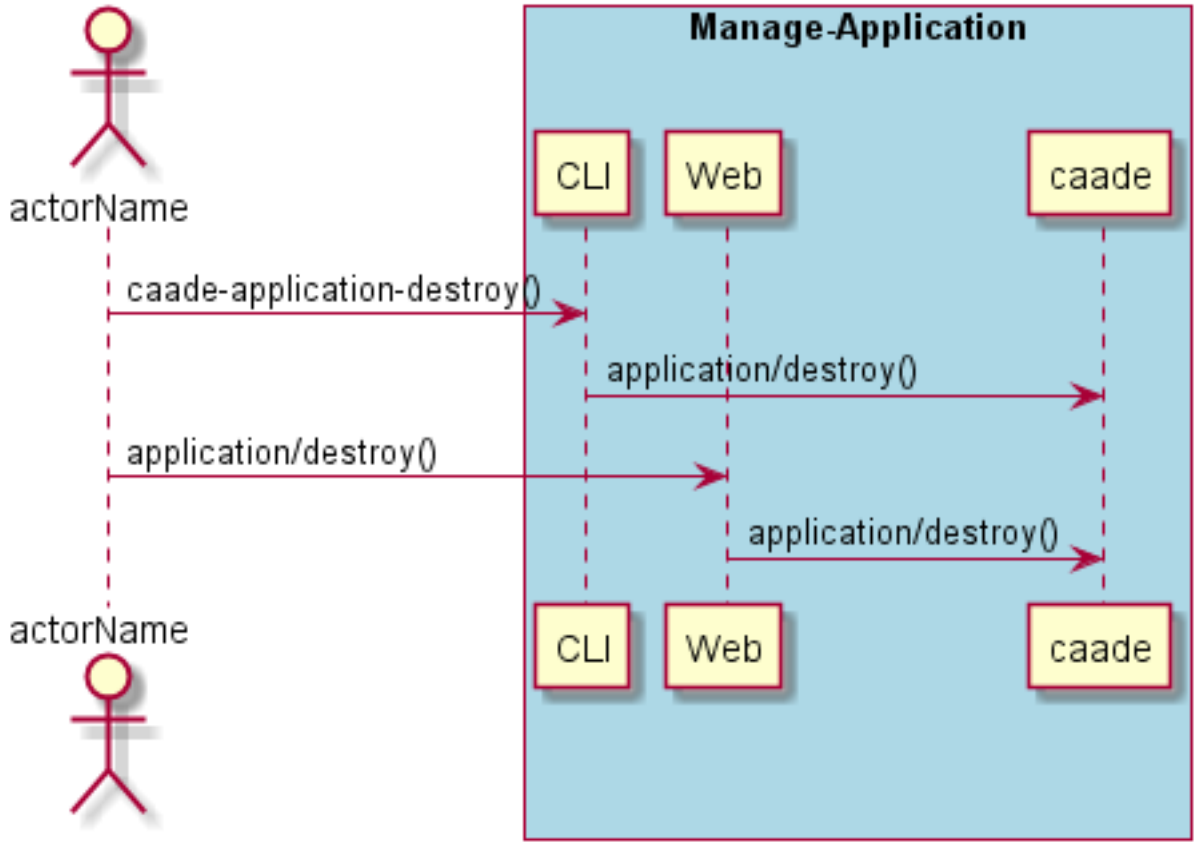
This is the RESTful interface for the scenario.

application/create

Name	Value	Description
parameter1	value1	Description1

Destroy Application

Destroy Application using CLI and Web Interface with . . . <parameters>



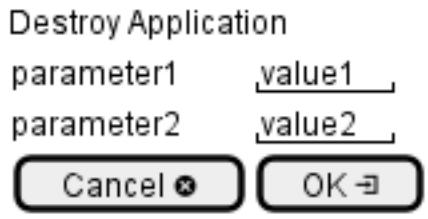
CLI

This is the command line interface for the Destroy Application Scenario.

```
# caade application destroy <parameters>
# caade application destroy exmaple
```

Web Interface

This is a mock up of the Web Interface for the Destroy Application Scenario.



REST

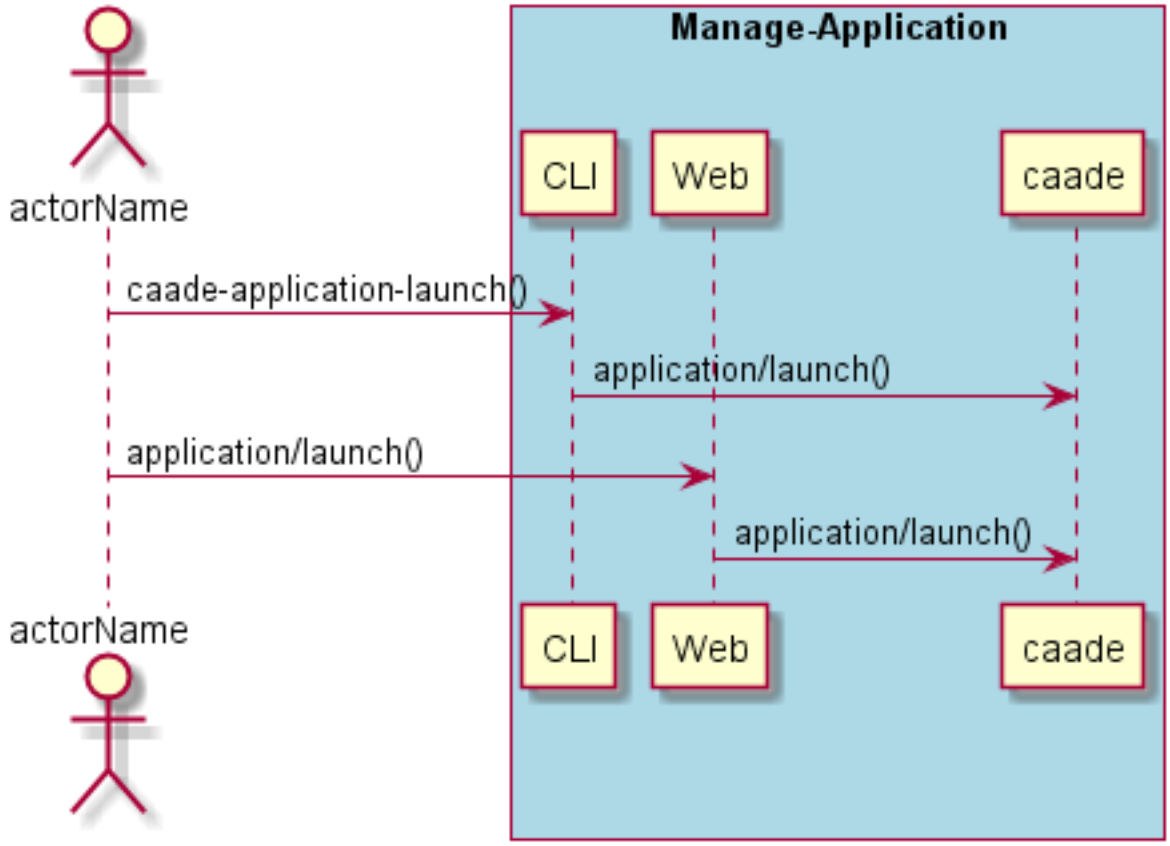
This is the RESTful interface for the scenario.

application/destroy

Name	Value	Description
parameter1	value1	Description1

Launch Application

Launch Application using CLI and Web Interface with ... <parameters>



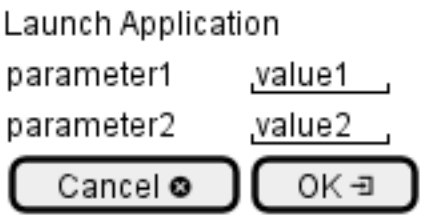
CLI

This is the command line interface for the Launch Application Scenario.

```
# caade application launch <parameters>
# caade application launch exmaple
```

Web Interface

This is a mock up of the Web Interface for the Launch Application Scenario.



REST

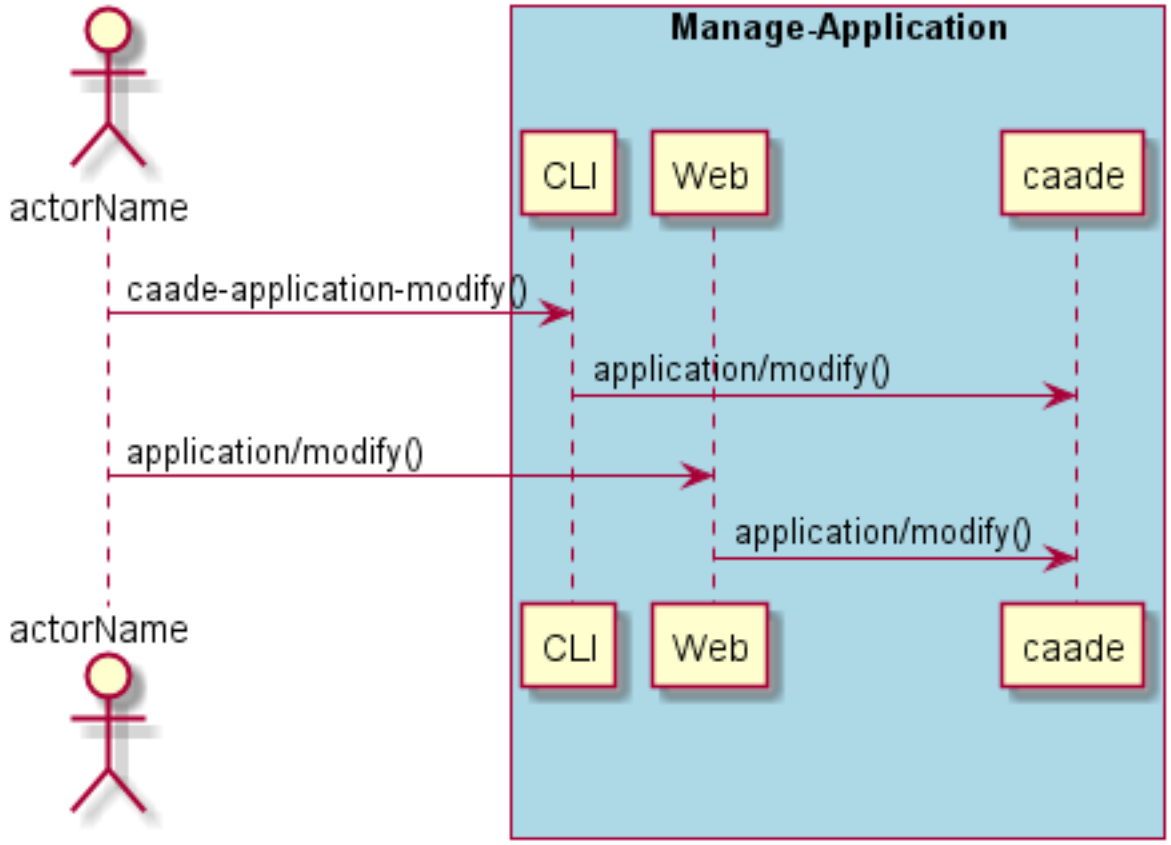
This is the RESTful interface for the scenario.

application/launch

Name	Value	Description
parameter1	value1	Description1

Modify Application

Modify Application using CLI and Web Interface with . . . <parameters>



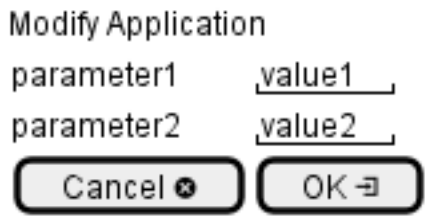
CLI

This is the command line interface for the Modify Application Scenario.

```
# caade application modify <parameters>
# caade application modify exmaple
```

Web Interface

This is a mock up of the Web Interface for the Modify Application Scenario.



REST

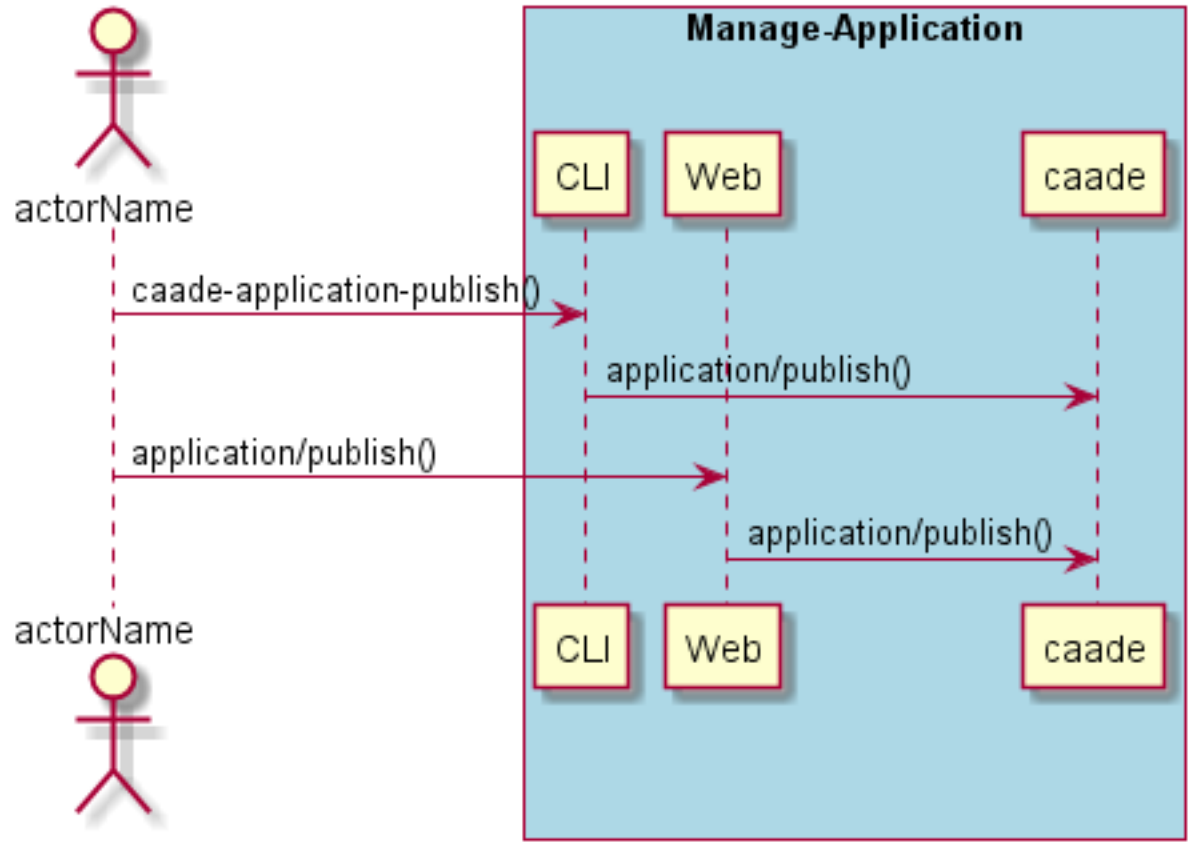
This is the RESTful interface for the scenario.

application/modify

Name	Value	Description
parameter1	value1	Description1

Publish Application

Publish Application using CLI and Web Interface with . . . <parameters>



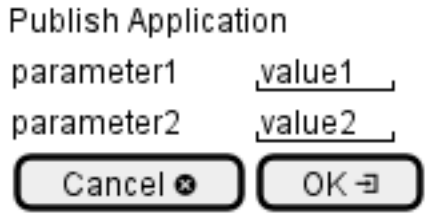
CLI

This is the command line interface for the Publish Application Scenario.

```
# caade application publish <parameters>
# caade application publish exmaple
```

Web Interface

This is a mock up of the Web Interface for the Publish Application Scenario.



REST

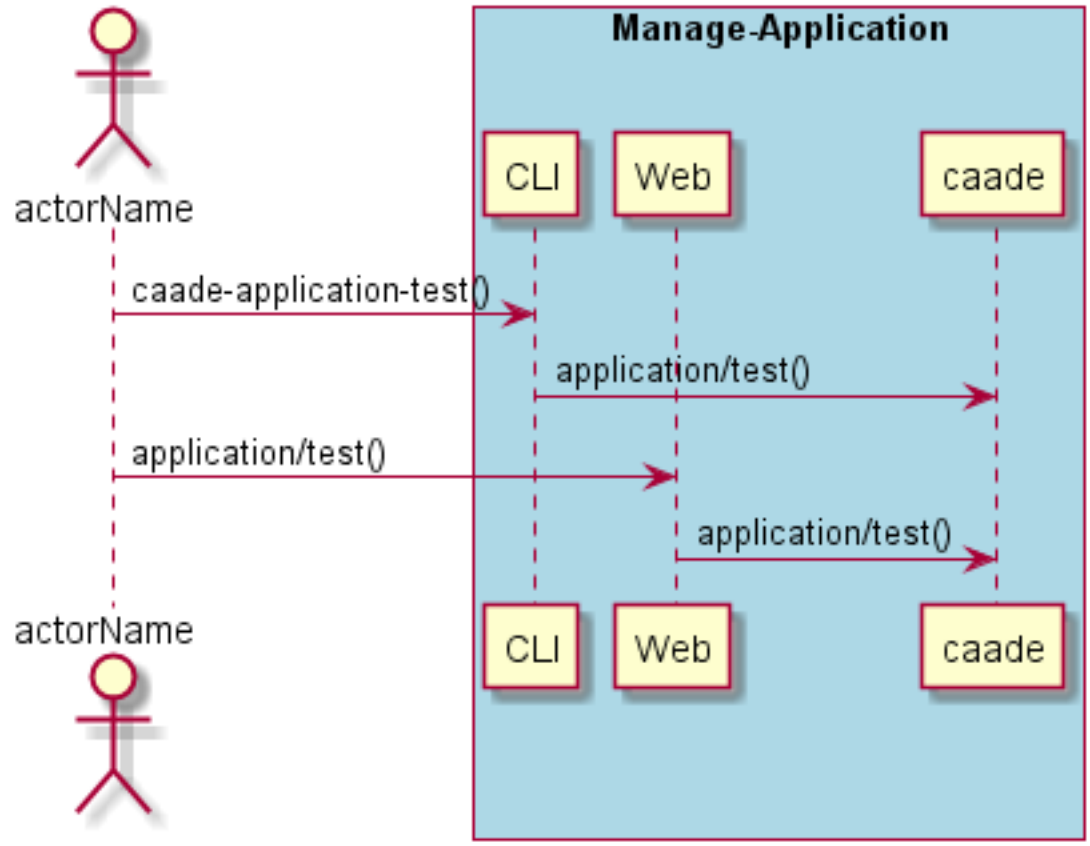
This is the RESTful interface for the scenario.

application/publish

Name	Value	Description
parameter1	value1	Description1

Test Application

Test Application using CLI and Web Interface with . . . <parameters>



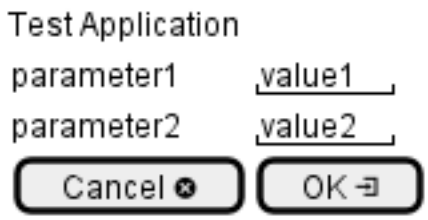
CLI

This is the command line interface for the Test Application Scenario.

```
# caade application test <parameters>
# caade application test exmaple
```

Web Interface

This is a mock up of the Web Interface for the Test Application Scenario.



REST

This is the RESTful interface for the scenario.

application/test

Name	Value	Description
parameter1	value1	Description1

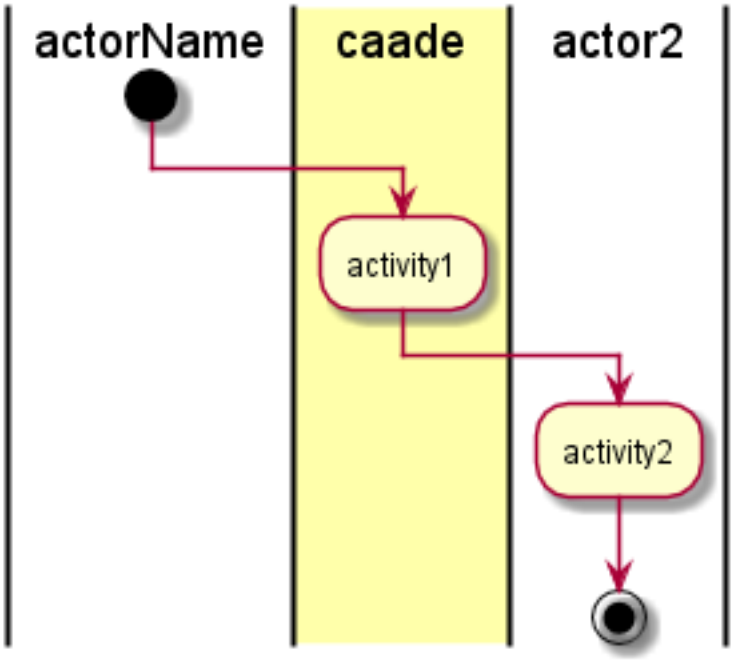
4.2 Manage Build

Add Description

4.2.1 Actors

- *DevOps*

4.2.2 Activities



- Activity from the diagram

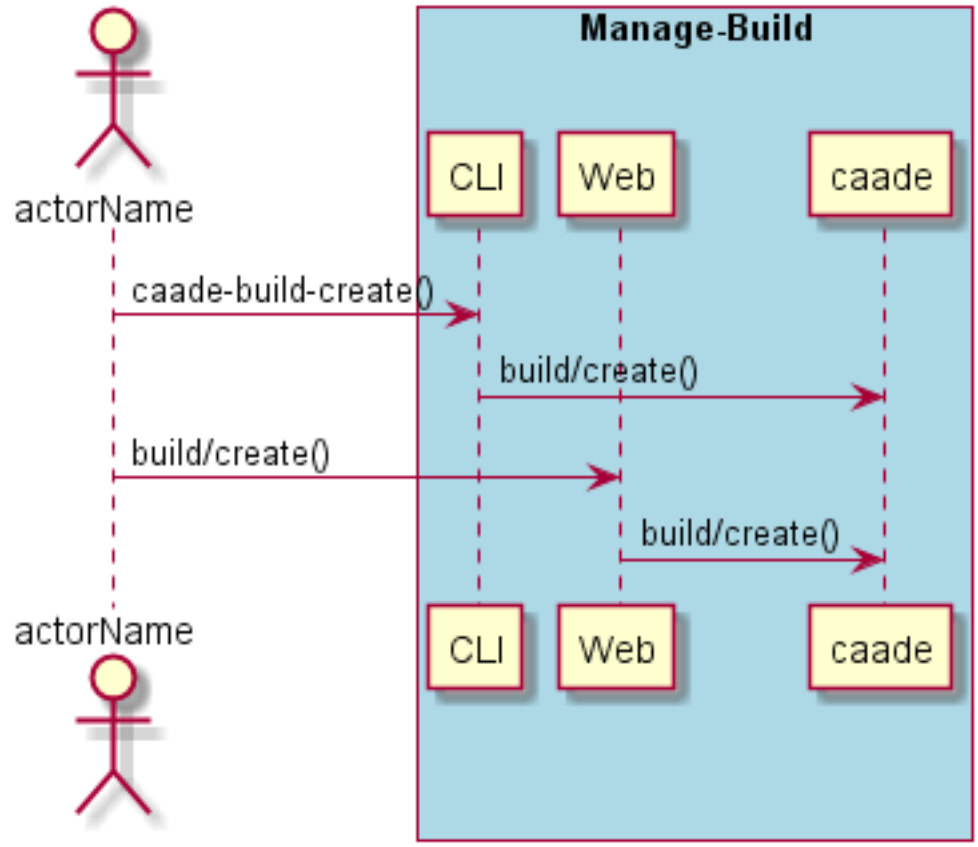
4.2.3 Systems Involved

- *Hybrid Cloud*

4.2.4 Detail Scenarios

Create Build

Create Build using CLI and Web Interface with . . . <parameters>



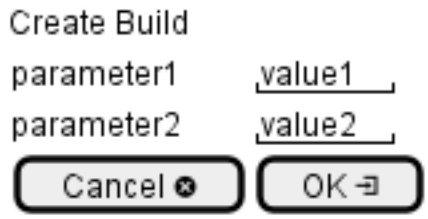
CLI

This is the command line interface for the Create Build Scenario.

```
# caade build create <parameters>
# caade build create exmaple
```

Web Interface

This is a mock up of the Web Interface for the Create Build Scenario.



REST

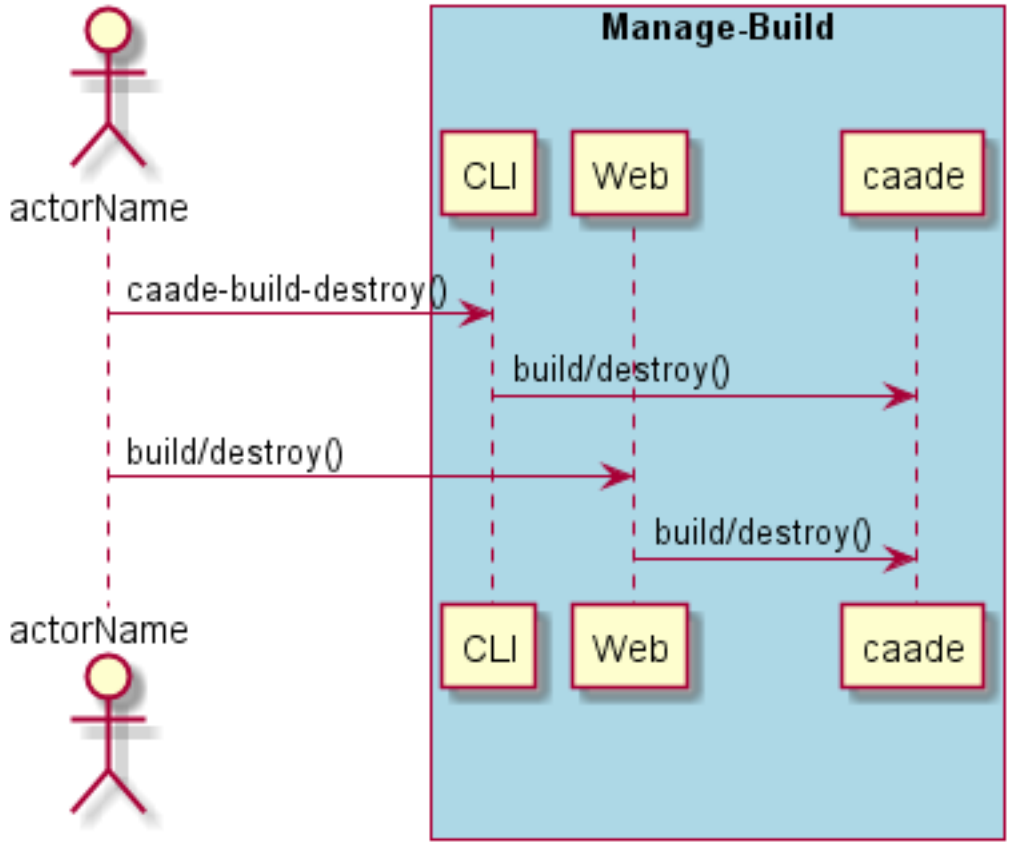
This is the RESTful interface for the scenario.

build/create

Name	Value	Description
parameter1	value1	Description1

Destroy Build

Destroy Build using CLI and Web Interface with . . . <parameters>



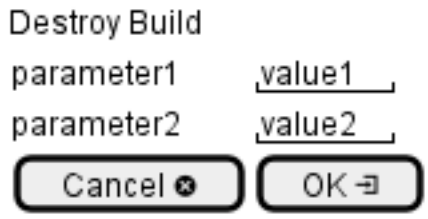
CLI

This is the command line interface for the Destroy Build Scenario.

```
# caade build destroy <parameters>
# caade build destroy exmaple
```

Web Interface

This is a mock up of the Web Interface for the Destroy Build Scenario.



REST

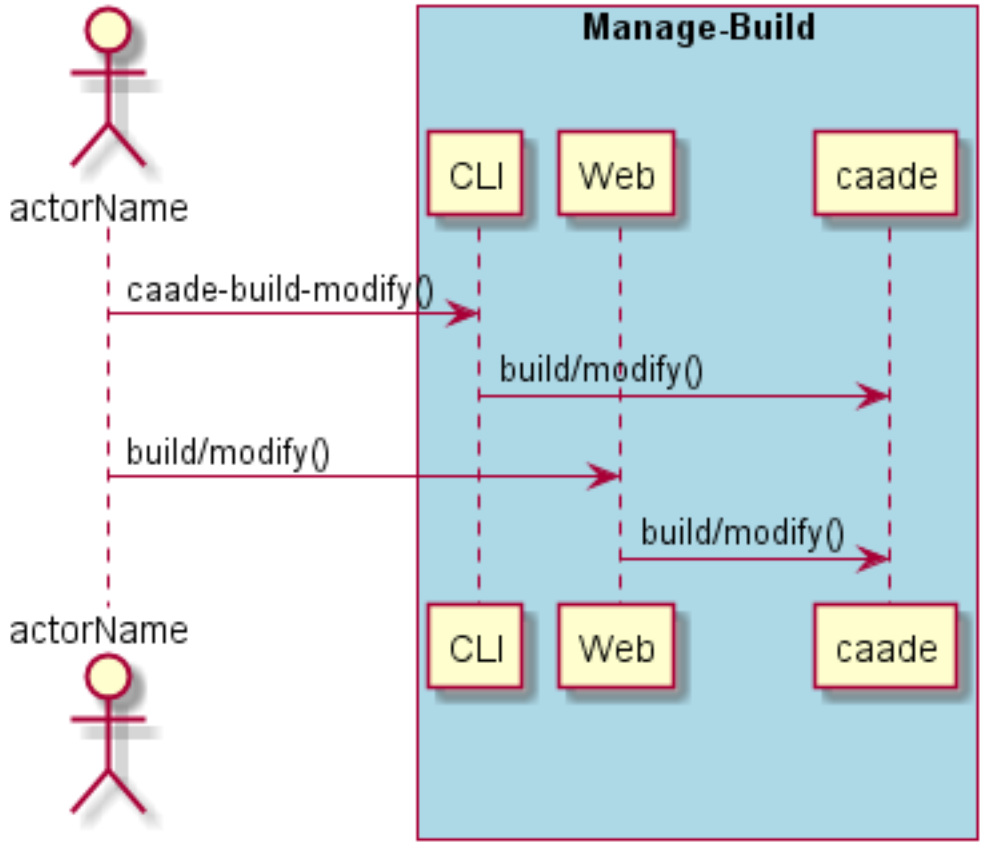
This is the RESTful interface for the scenario.

build/destroy

Name	Value	Description
parameter1	value1	Description1

Modify Build

Modify Build using CLI and Web Interface with ... <parameters>



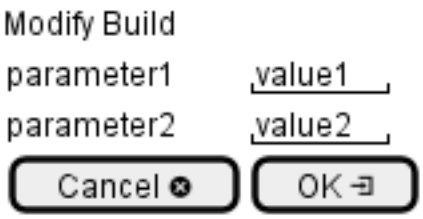
CLI

This is the command line interface for the Modify Build Scenario.

```
# caade build modify <parameters>
# caade build modify exmaple
```

Web Interface

This is a mock up of the Web Interface for the Modify Build Scenario.



REST

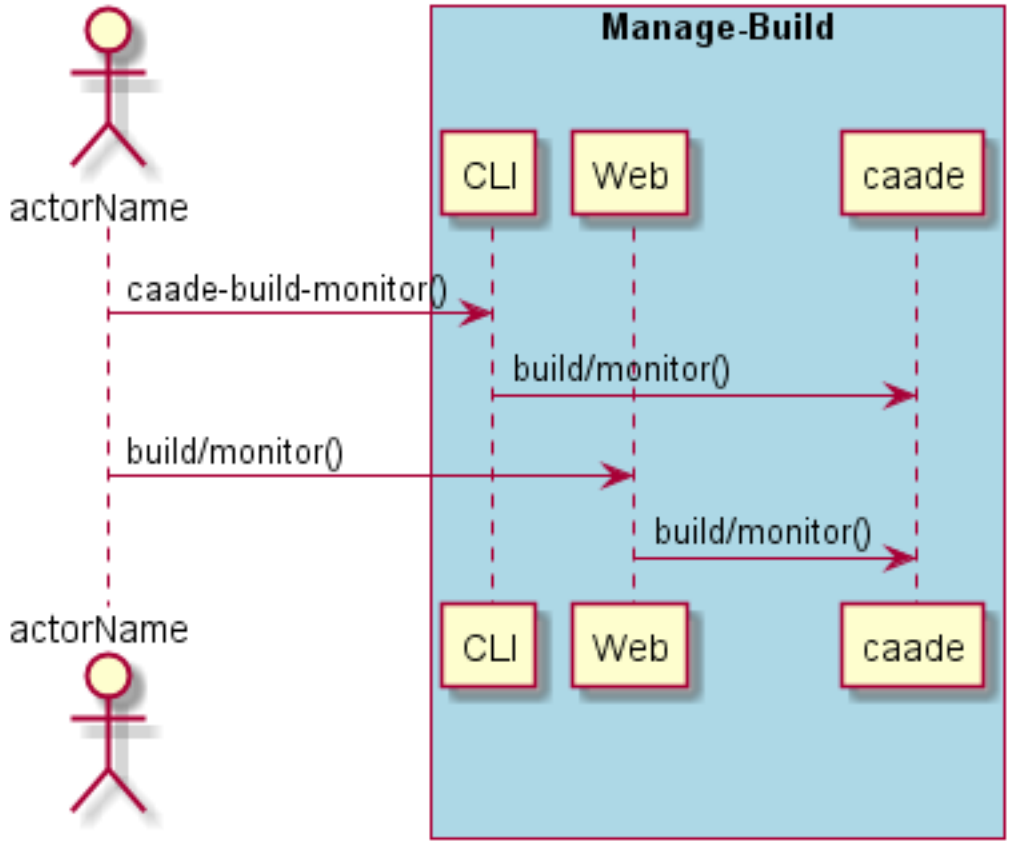
This is the RESTful interface for the scenario.

build/modify

Name	Value	Description
parameter1	value1	Description1

Monitor Build

Monitor Build using CLI and Web Interface with . . . <parameters>



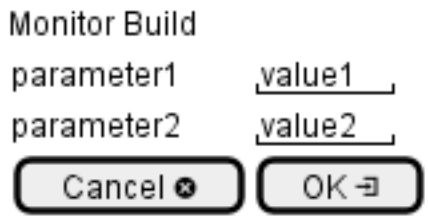
CLI

This is the command line interface for the Monitor Build Scenario.

```
# caade build monitor <parameters>
# caade build monitor exmaple
```

Web Interface

This is a mock up of the Web Interface for the Monitor Build Scenario.



REST

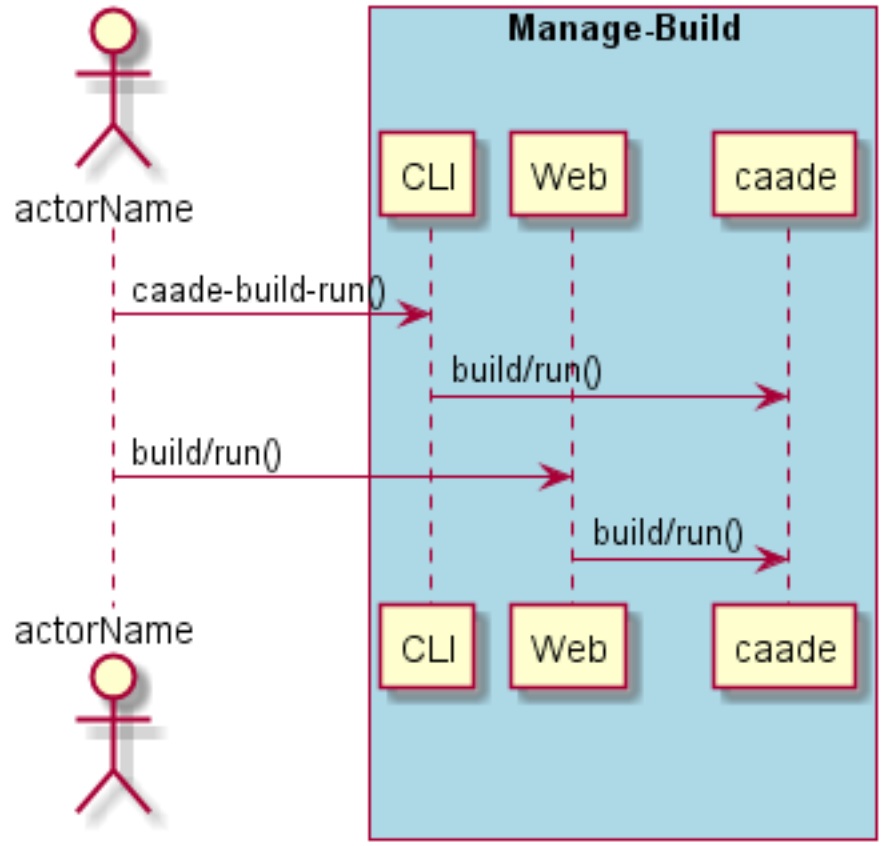
This is the RESTful interface for the scenario.

build/monitor

Name	Value	Description
parameter1	value1	Description1

Run Build

Run Build using CLI and Web Interface with . . . <parameters>



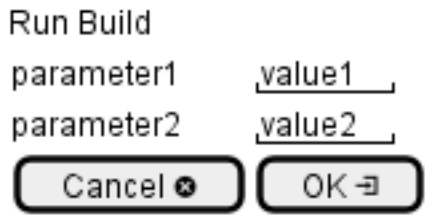
CLI

This is the command line interface for the Run Build Scenario.

```
# caade build run <parameters>
# caade build run exmaple
```

Web Interface

This is a mock up of the Web Interface for the Run Build Scenario.



REST

This is the RESTful interface for the scenario.

build/run

Name	Value	Description
parameter1	value1	Description1

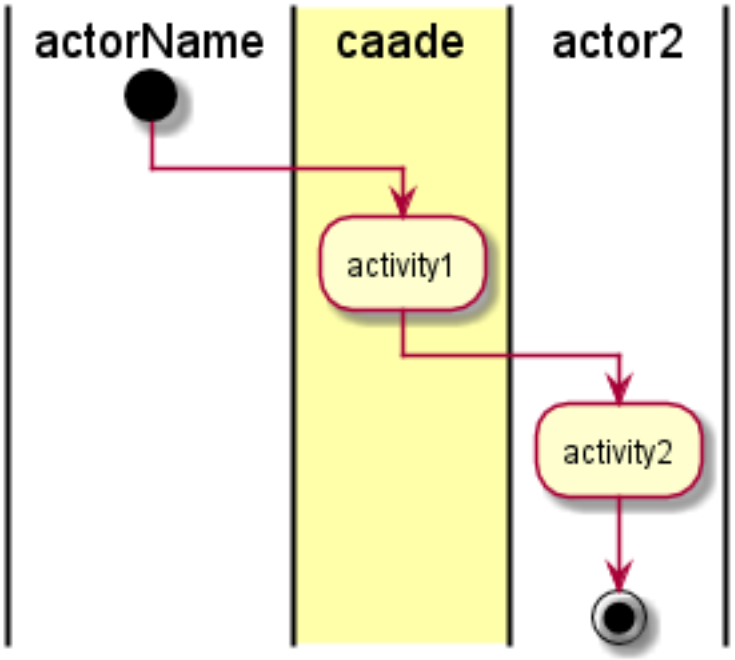
4.3 Manage Code

Add Description

4.3.1 Actors

- *DevOps*

4.3.2 Activities



- Activity from the diagram

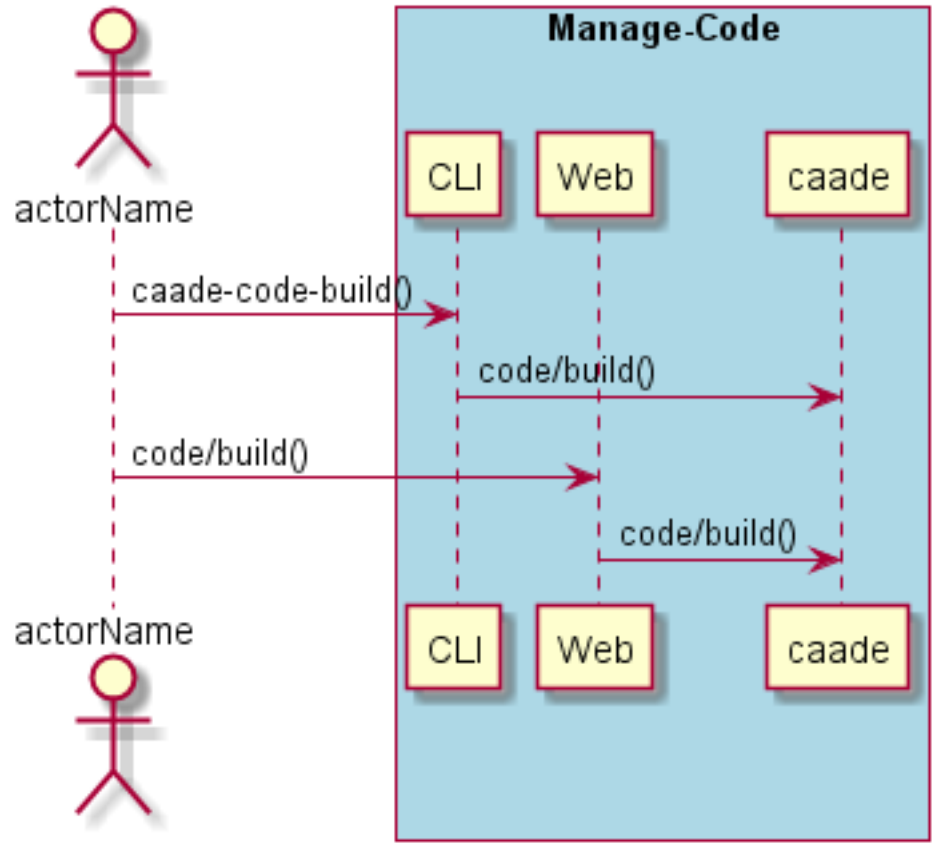
4.3.3 Systems Involved

- *Hybrid Cloud*

4.3.4 Detail Scenarios

Build Code

Build Code using CLI and Web Interface with . . . <parameters>



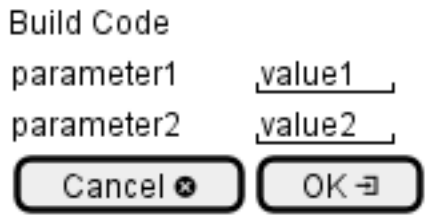
CLI

This is the command line interface for the Build Code Scenario.

```
# caade code build <parameters>
# caade code build exmaple
```

Web Interface

This is a mock up of the Web Interface for the Build Code Scenario.



REST

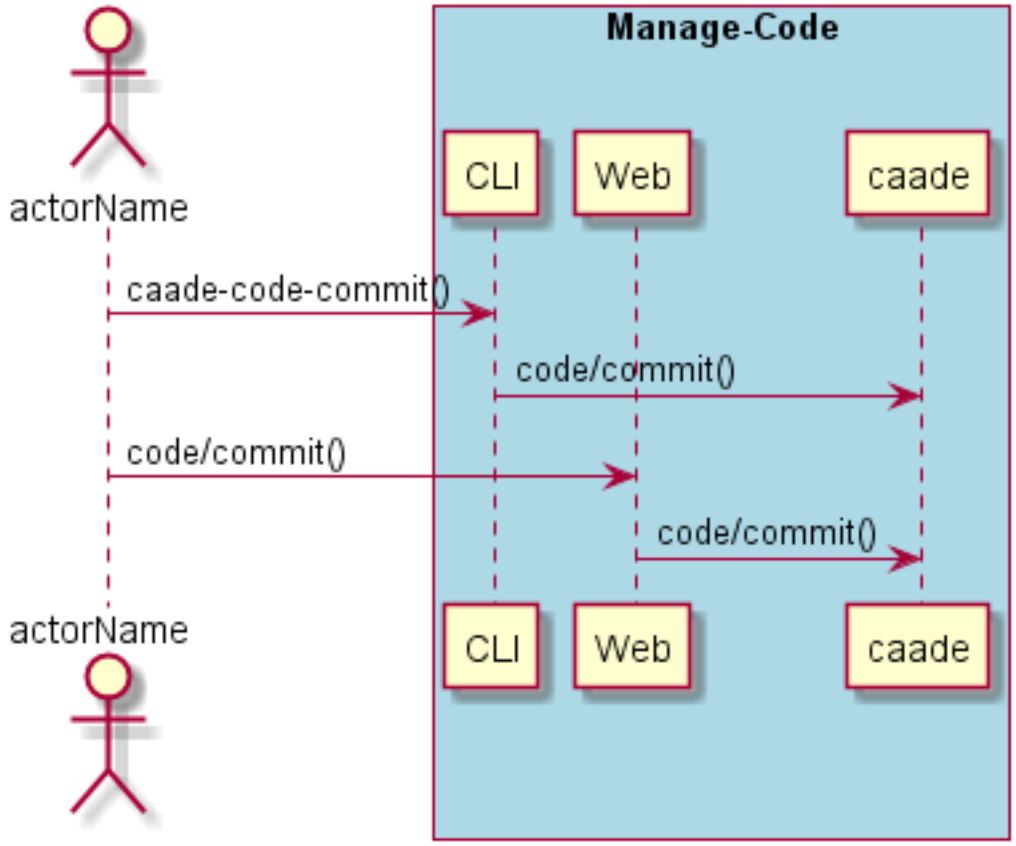
This is the RESTful interface for the scenario.

code/build

Name	Value	Description
parameter1	value1	Description1

Commit Code

Commit Code using CLI and Web Interface with . . . <parameters>



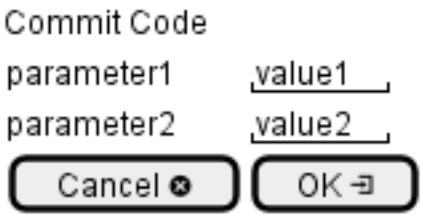
CLI

This is the command line interface for the Commit Code Scenario.

```
# caade code commit <parameters>
# caade code commit exmaple
```

Web Interface

This is a mock up of the Web Interface for the Commit Code Scenario.



REST

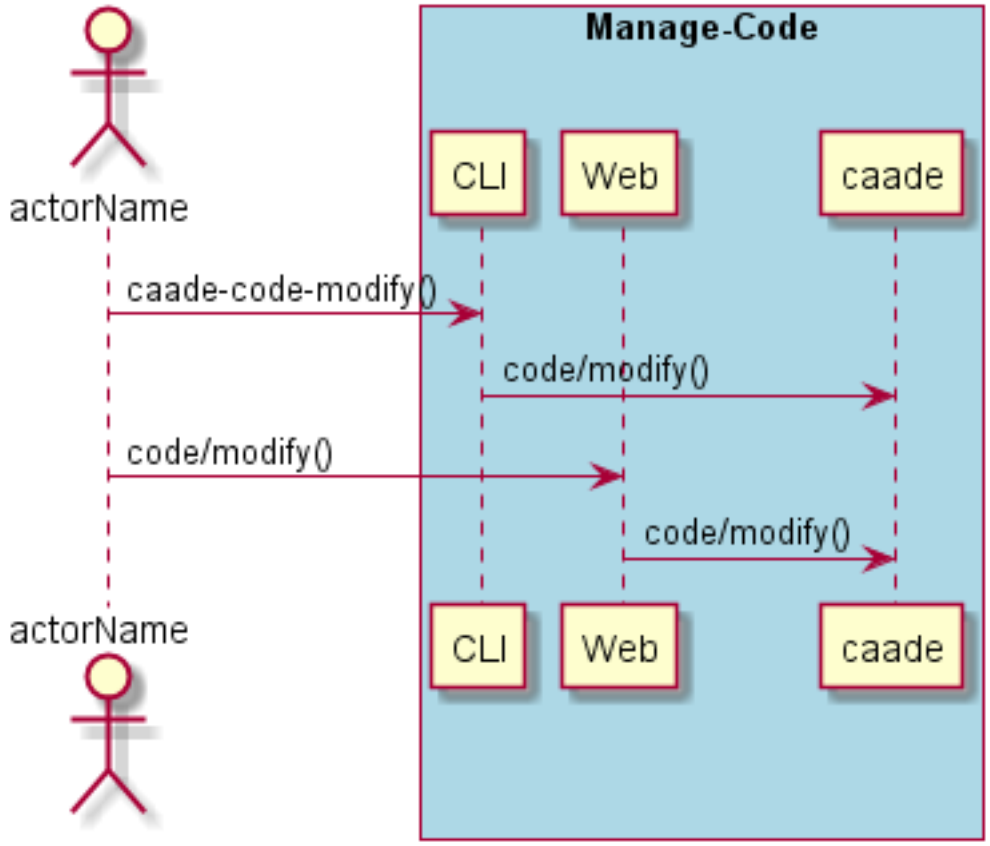
This is the RESTful interface for the scenario.

code/commit

Name	Value	Description
parameter1	value1	Description1

Modify Code

Modify Code using CLI and Web Interface with ... <parameters>



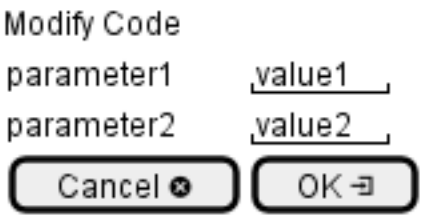
CLI

This is the command line interface for the Modify Code Scenario.

```
# caade code modify <parameters>
# caade code modify exmaple
```

Web Interface

This is a mock up of the Web Interface for the Modify Code Scenario.



REST

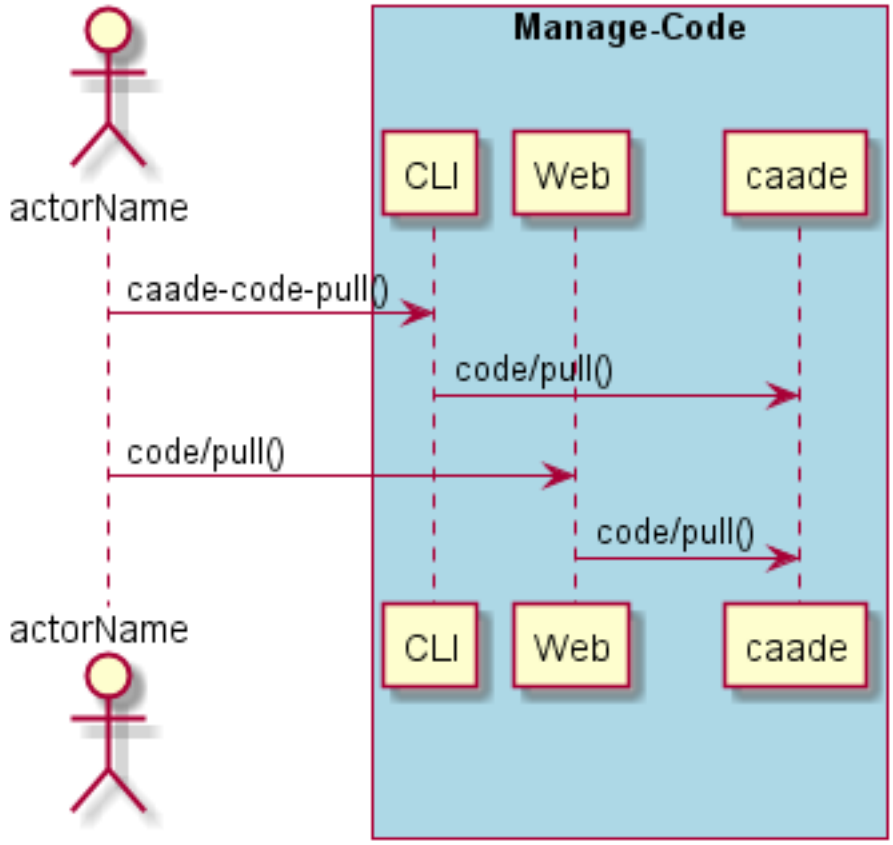
This is the RESTful interface for the scenario.

code/modify

Name	Value	Description
parameter1	value1	Description1

Pull Code

Pull Code using CLI and Web Interface with ... <parameters>



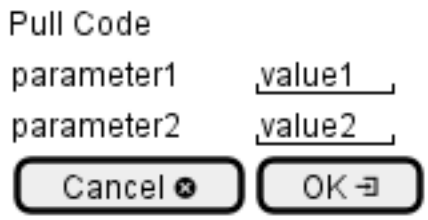
CLI

This is the command line interface for the Pull Code Scenario.

```
# caade code pull <parameters>
# caade code pull exmaple
```

Web Interface

This is a mock up of the Web Interface for the Pull Code Scenario.



REST

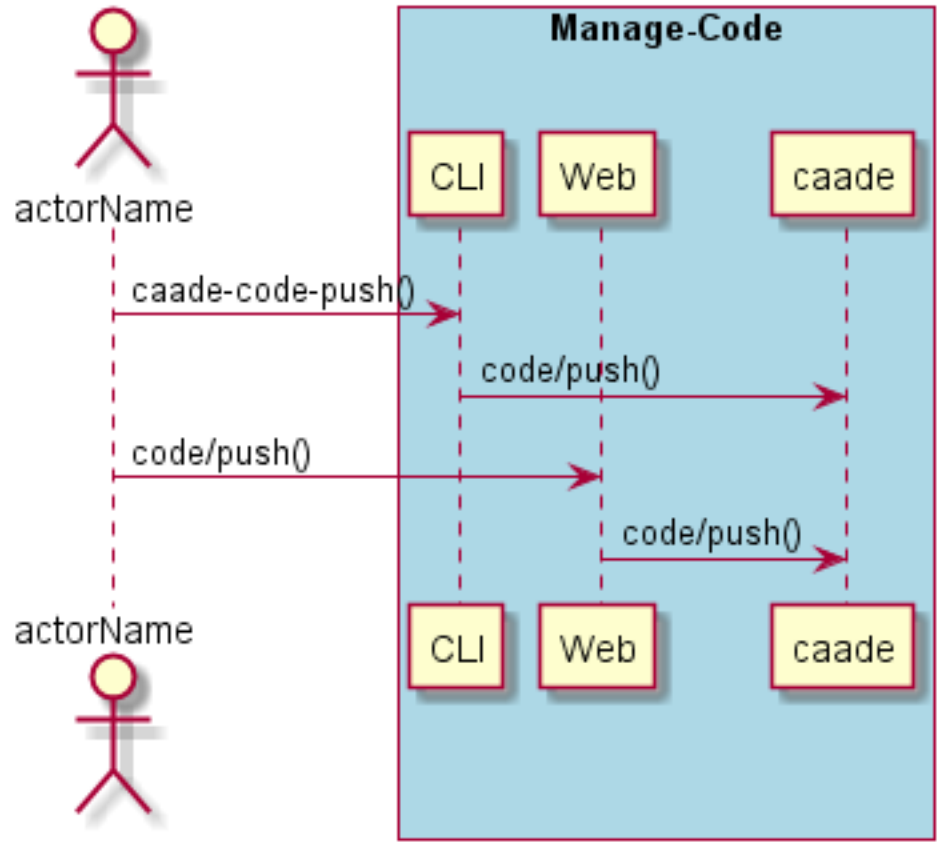
This is the RESTful interface for the scenario.

code/pull

Name	Value	Description
parameter1	value1	Description1

Push Code

Push Code using CLI and Web Interface with … <parameters>



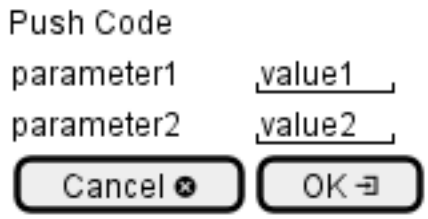
CLI

This is the command line interface for the Push Code Scenario.

```
# caade code push <parameters>
# caade code push exmaple
```

Web Interface

This is a mock up of the Web Interface for the Push Code Scenario.



REST

This is the RESTful interface for the scenario.

code/push

Name	Value	Description
parameter1	value1	Description1

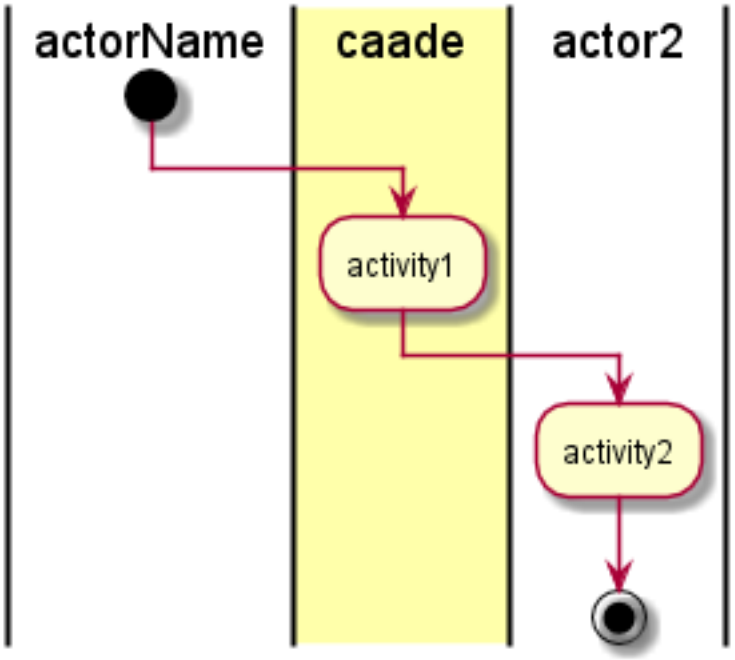
4.4 Manage Environment

Add Description

4.4.1 Actors

- *DevOps*

4.4.2 Activities



- Activity from the diagram

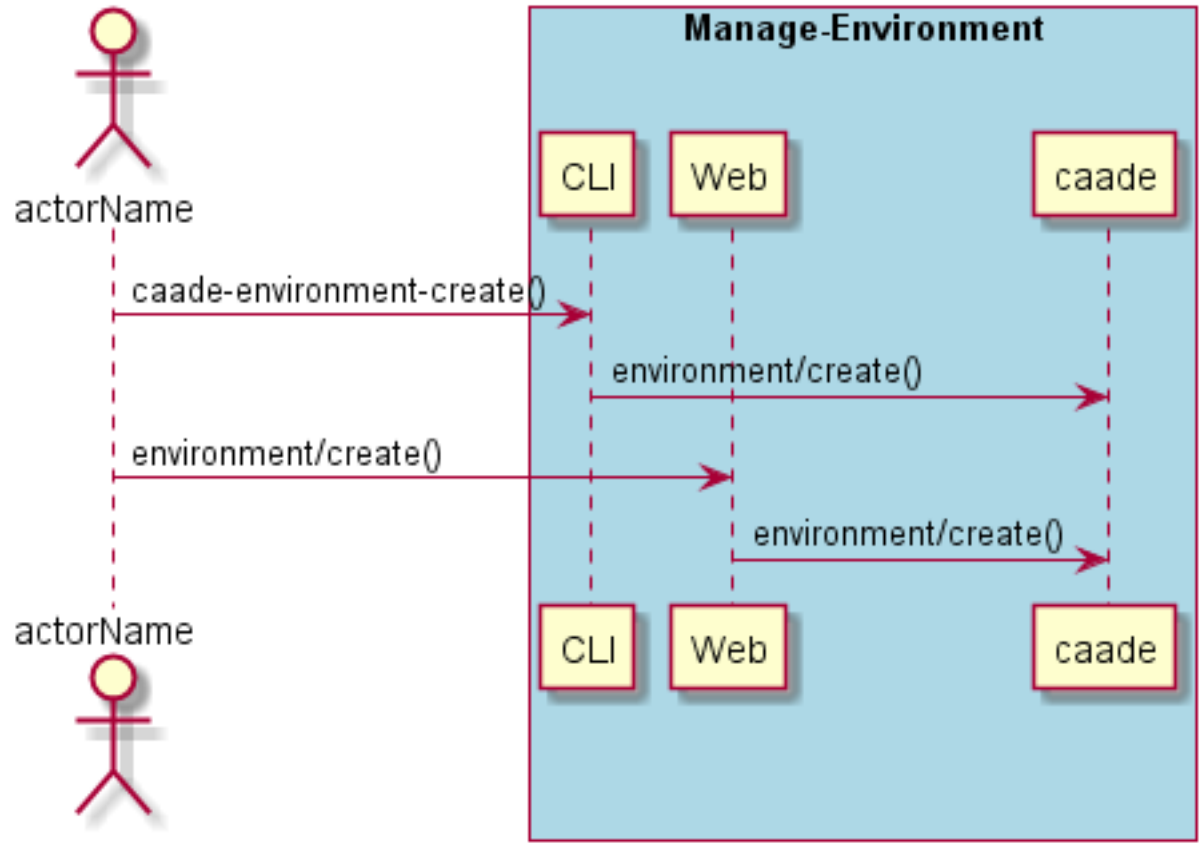
4.4.3 Systems Involved

- *Hybrid Cloud*

4.4.4 Detail Scenarios

Create Environment

Create Environment using CLI and Web Interface with . . . <parameters>



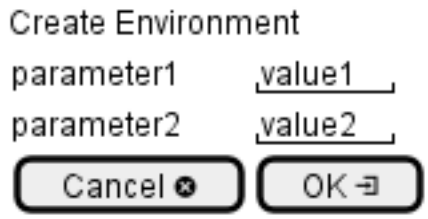
CLI

This is the command line interface for the Create Environment Scenario.

```
# caade environment create <parameters>
# caade environment create exmaple
```

Web Interface

This is a mock up of the Web Interface for the Create Environment Scenario.



REST

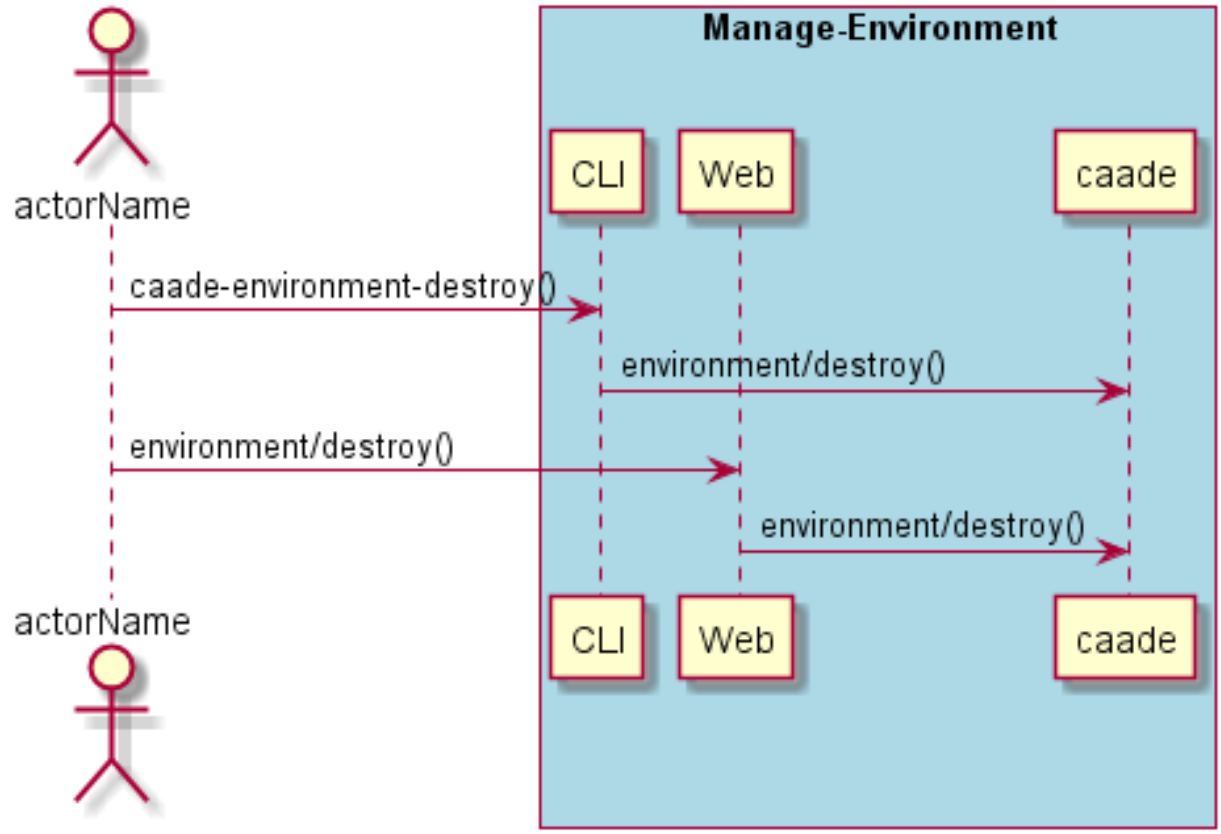
This is the RESTful interface for the scenario.

environment/create

Name	Value	Description
parameter1	value1	Description1

Destroy Environment

Destroy Environment using CLI and Web Interface with . . . <parameters>



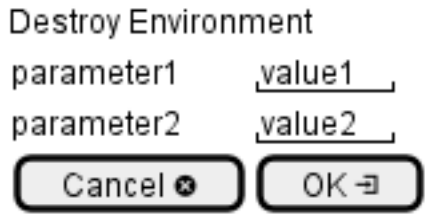
CLI

This is the command line interface for the Destroy Environment Scenario.

```
# caade environment destroy <parameters>
# caade environment destroy exmaple
```

Web Interface

This is a mock up of the Web Interface for the Destroy Environment Scenario.



REST

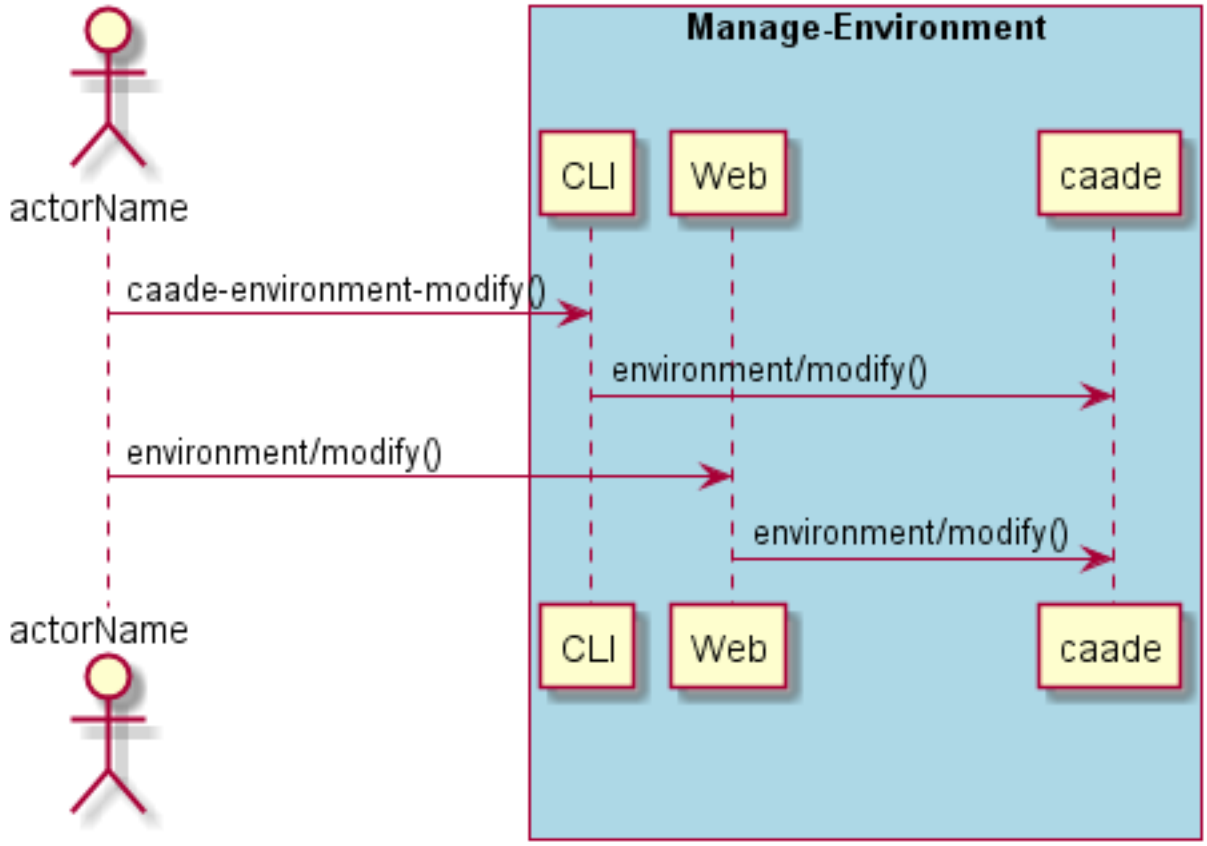
This is the RESTful interface for the scenario.

environment/destroy

Name	Value	Description
parameter1	value1	Description1

Modify Environment

Modify Environment using CLI and Web Interface with . . . <parameters>



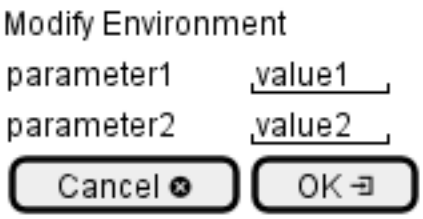
CLI

This is the command line interface for the Modify Environment Scenario.

```
# caade environment modify <parameters>
# caade environment modify exmaple
```

Web Interface

This is a mock up of the Web Interface for the Modify Environment Scenario.



REST

This is the RESTful interface for the scenario.

environment/modify

Name	Value	Description
parameter1	value1	Description1

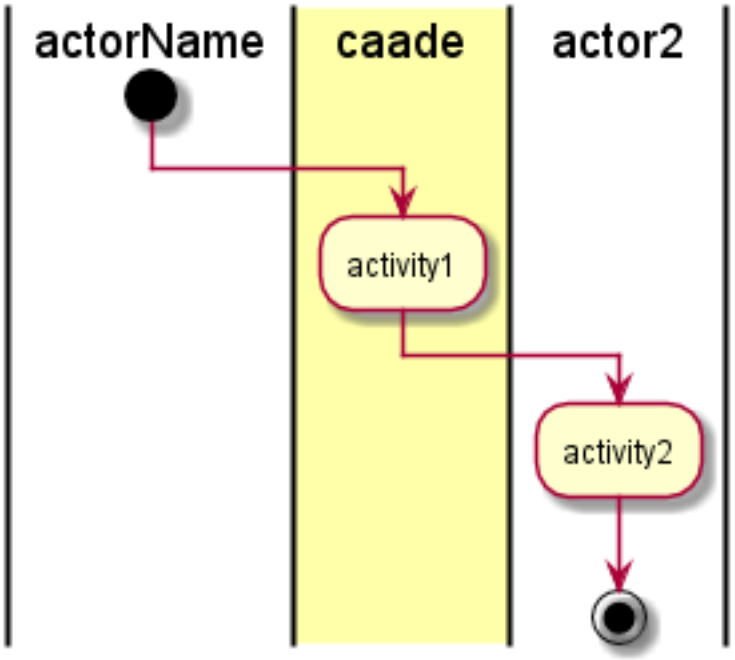
4.5 Manage Pipeline

Add Description

4.5.1 Actors

- *DevOps*

4.5.2 Activities



- Activity from the diagram

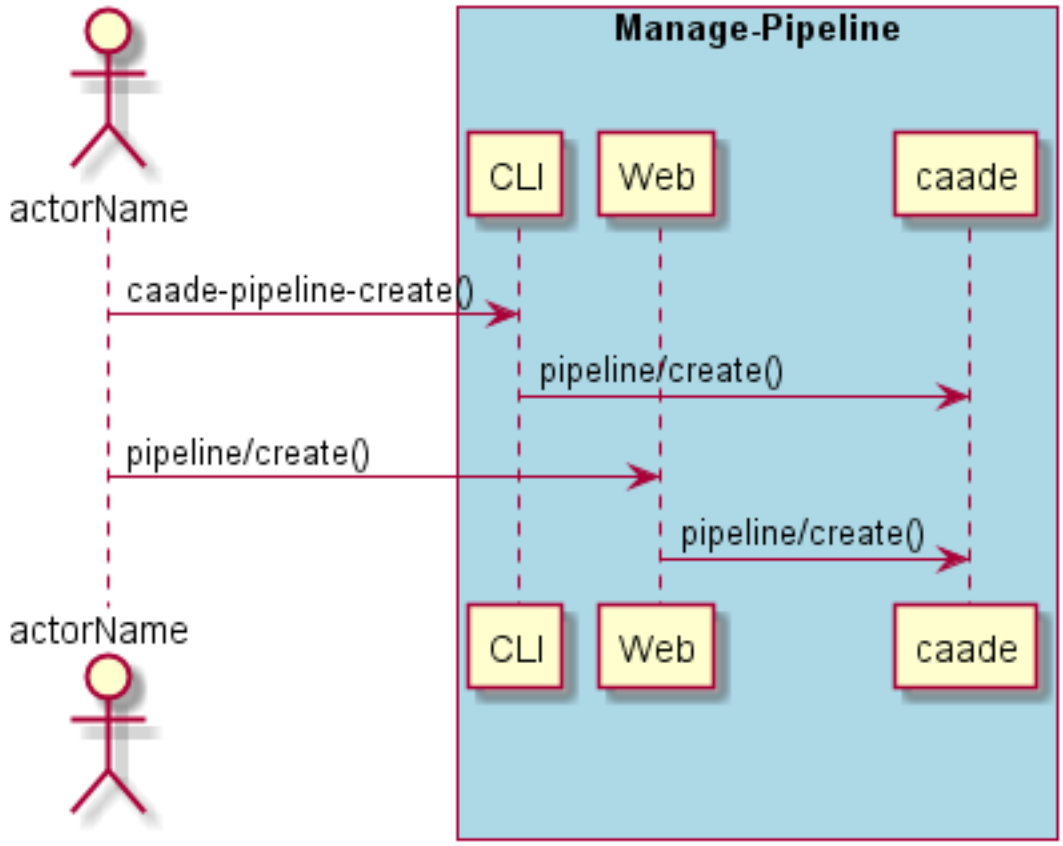
4.5.3 Systems Involved

- *Hybrid Cloud*

4.5.4 Detail Scenarios

Create Pipeline

Create Pipeline using CLI and Web Interface with ... <parameters>



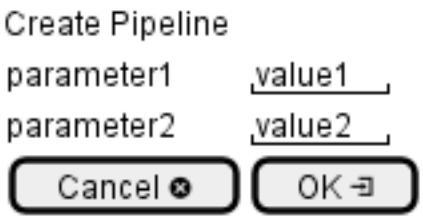
CLI

This is the command line interface for the Create Pipeline Scenario.

```
# caade pipeline create <parameters>
# caade pipeline create exmaple
```

Web Interface

This is a mock up of the Web Interface for the Create Pipeline Scenario.



REST

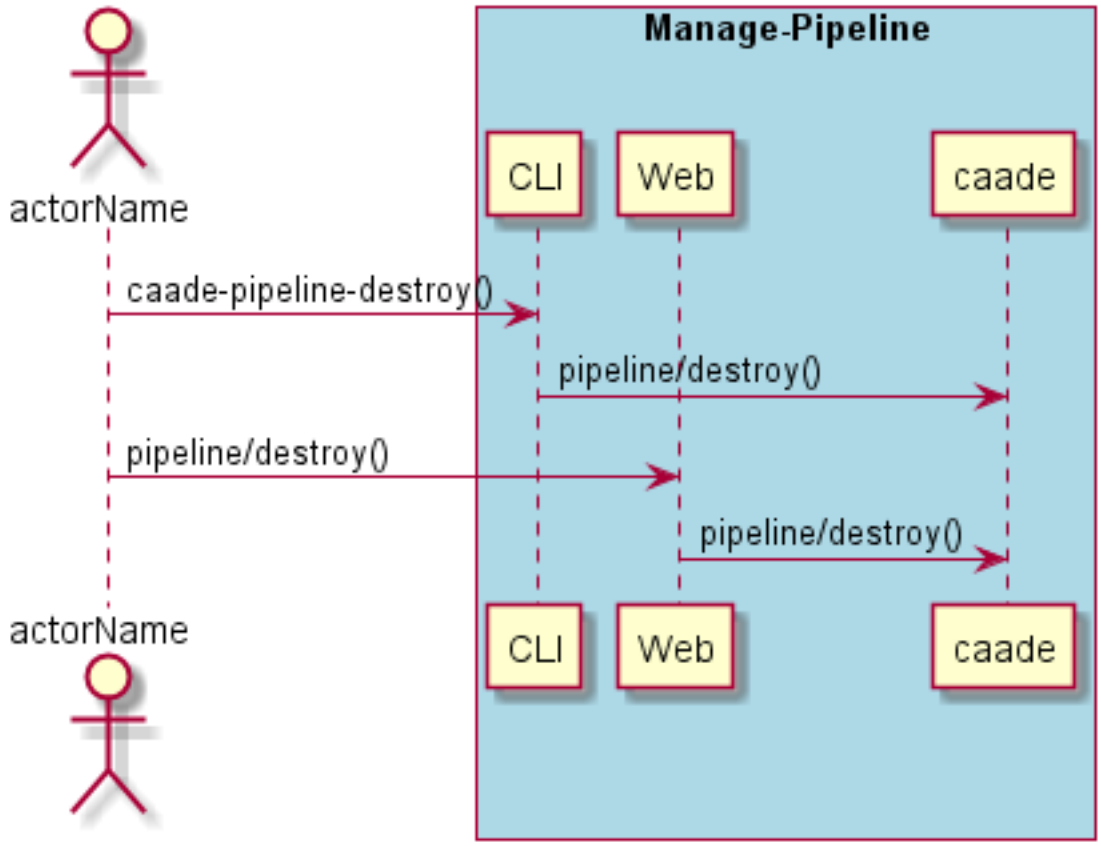
This is the RESTful interface for the scenario.

pipeline/create

Name	Value	Description
parameter1	value1	Description1

Destroy Pipeline

Destroy Pipeline using CLI and Web Interface with . . . <parameters>



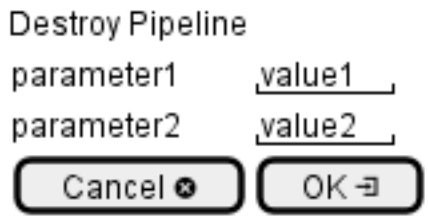
CLI

This is the command line interface for the Destroy Pipeline Scenario.

```
# caade pipeline destroy <parameters>
# caade pipeline destroy exmaple
```

Web Interface

This is a mock up of the Web Interface for the Destroy Pipeline Scenario.



REST

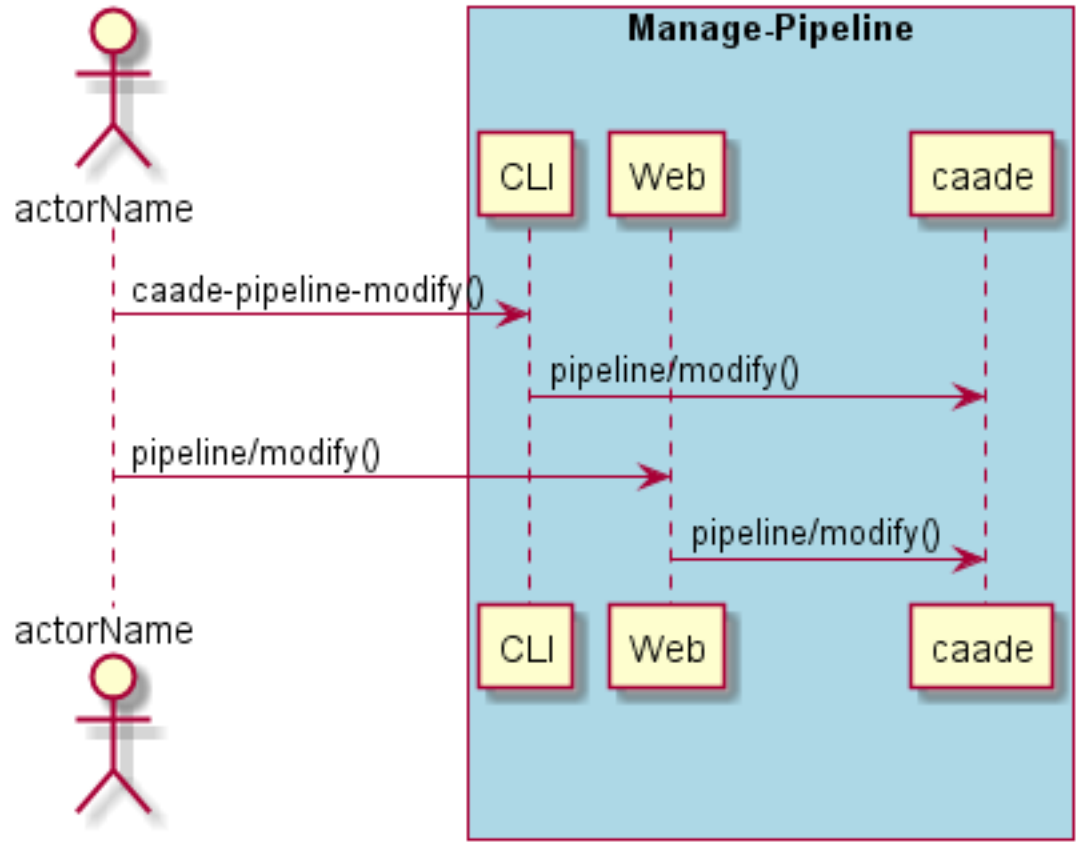
This is the RESTful interface for the scenario.

pipeline/destroy

Name	Value	Description
parameter1	value1	Description1

Modify Pipeline

Modify Pipeline using CLI and Web Interface with ... <parameters>



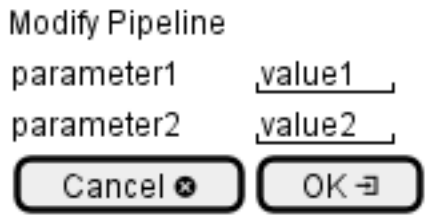
CLI

This is the command line interface for the Modify Pipeline Scenario.

```
# caade pipeline modify <parameters>
# caade pipeline modify exmaple
```

Web Interface

This is a mock up of the Web Interface for the Modify Pipeline Scenario.



REST

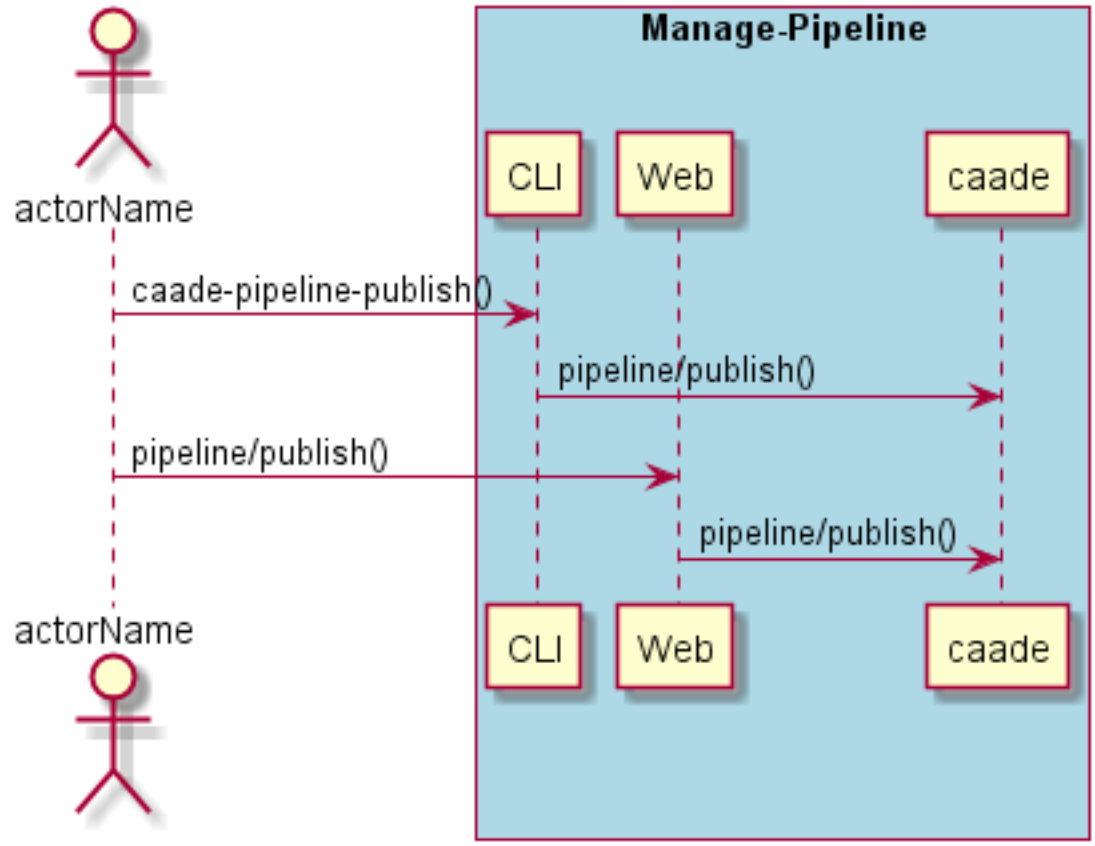
This is the RESTful interface for the scenario.

pipeline/modify

Name	Value	Description
parameter1	value1	Description1

Publish Pipeline

Publish Pipeline using CLI and Web Interface with . . . <parameters>



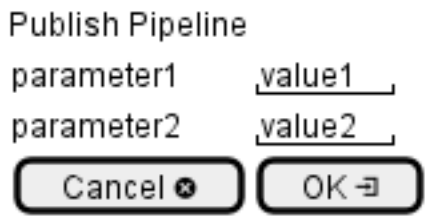
CLI

This is the command line interface for the Publish Pipeline Scenario.

```
# caade pipeline publish <parameters>
# caade pipeline publish exmaple
```

Web Interface

This is a mock up of the Web Interface for the Publish Pipeline Scenario.



REST

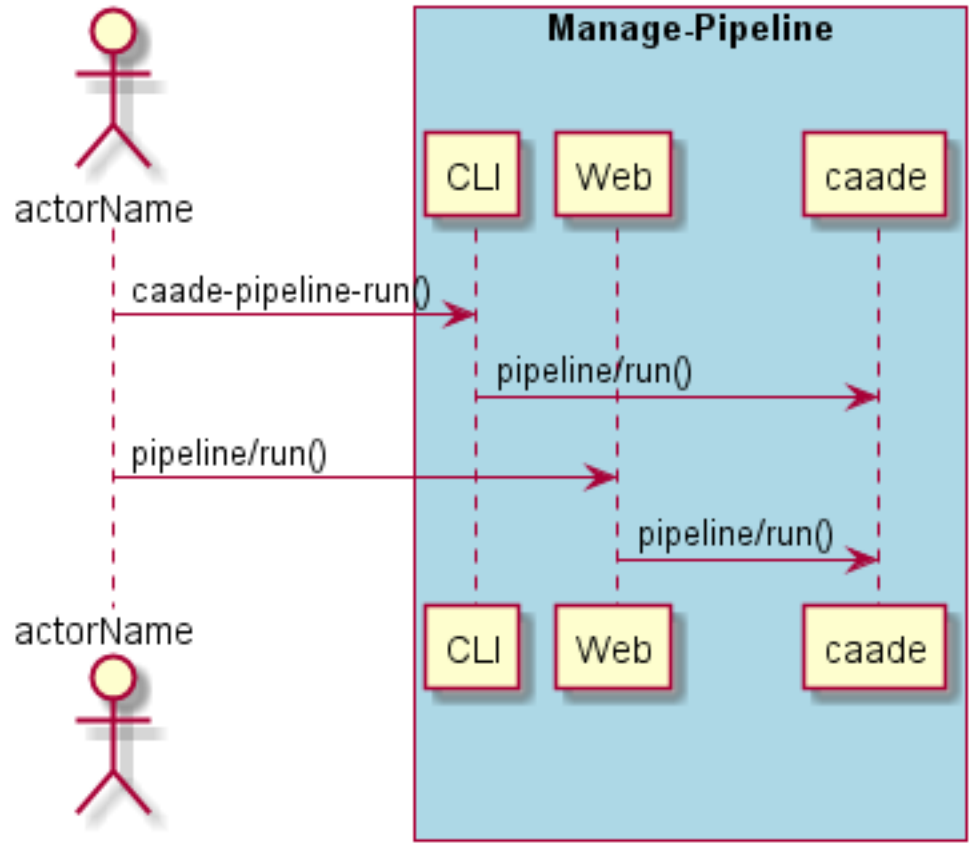
This is the RESTful interface for the scenario.

pipeline/publish

Name	Value	Description
parameter1	value1	Description1

Run Pipeline

Run Pipeline using CLI and Web Interface with . . . <parameters>



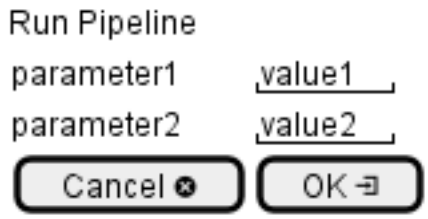
CLI

This is the command line interface for the Run Pipeline Scenario.

```
# caade pipeline run <parameters>
# caade pipeline run exmaple
```

Web Interface

This is a mock up of the Web Interface for the Run Pipeline Scenario.



REST

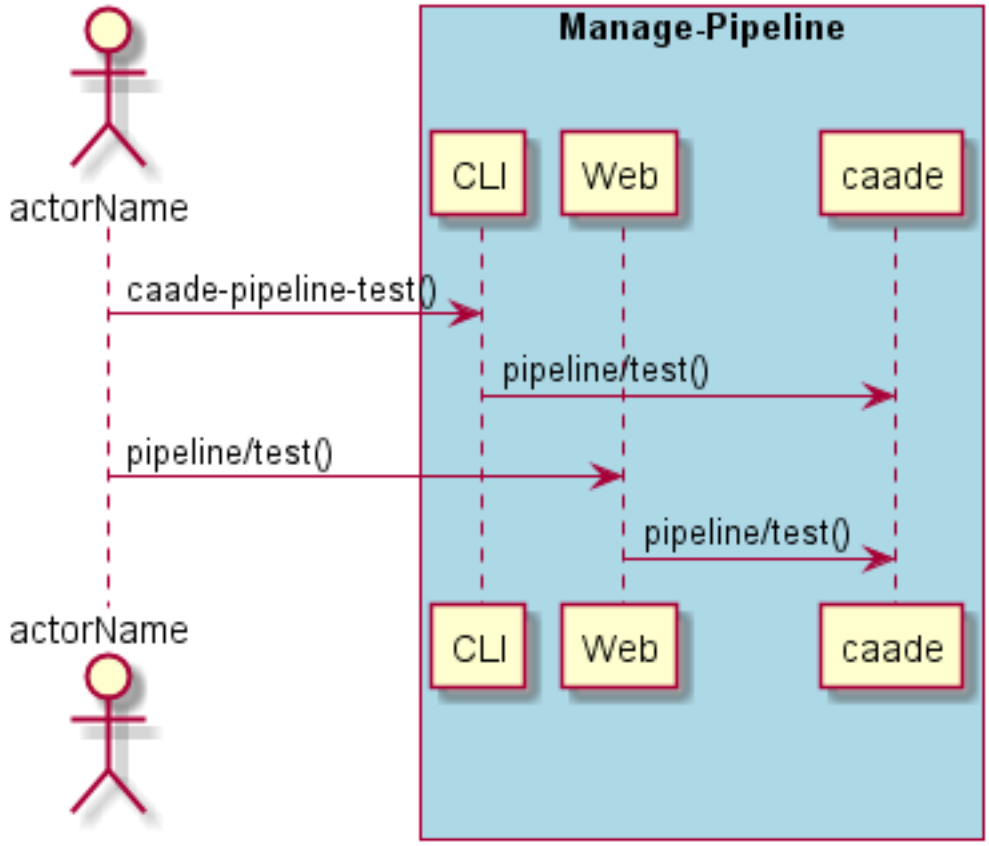
This is the RESTful interface for the scenario.

pipeline/run

Name	Value	Description
parameter1	value1	Description1

Test Pipeline

Test Pipeline using CLI and Web Interface with ... <parameters>



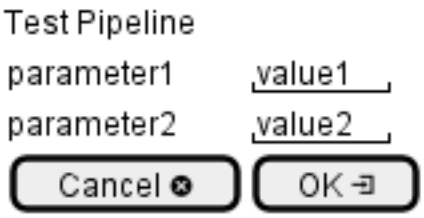
CLI

This is the command line interface for the Test Pipeline Scenario.

```
# caade pipeline test <parameters>
# caade pipeline test exmaple
```

Web Interface

This is a mock up of the Web Interface for the Test Pipeline Scenario.



REST

This is the RESTful interface for the scenario.

pipeline/test

Name	Value	Description
parameter1	value1	Description1

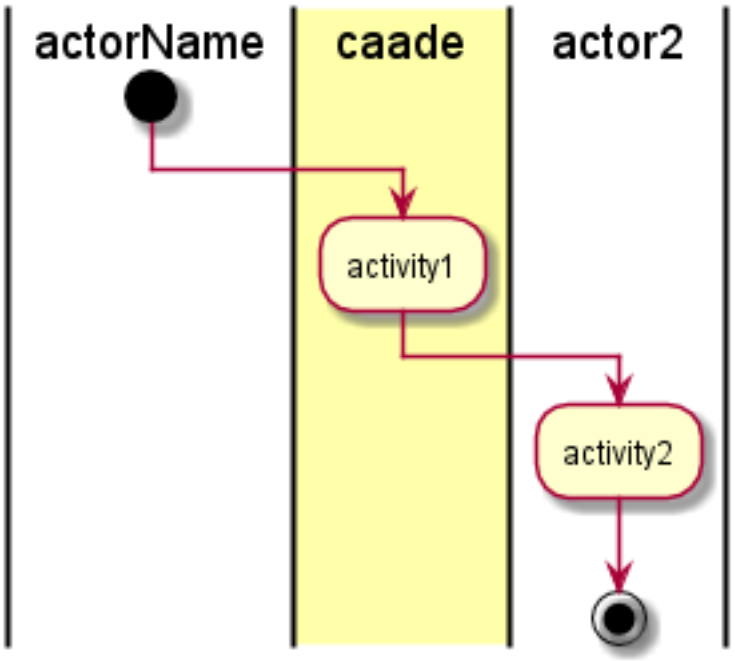
4.6 Manage Project

Add Description

4.6.1 Actors

- *DevOps*

4.6.2 Activities



- Activity from the diagram

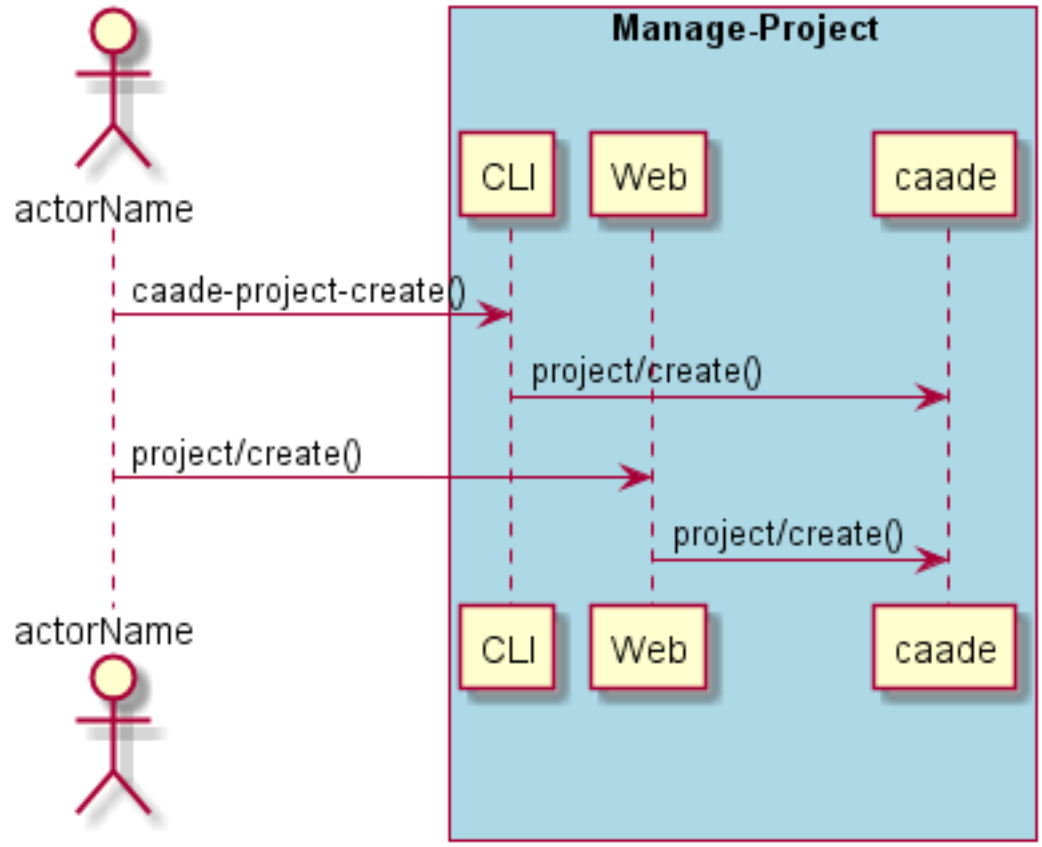
4.6.3 Systems Involved

- *Hybrid Cloud*

4.6.4 Detail Scenarios

Create Project

Create Project using CLI and Web Interface with . . . <parameters>



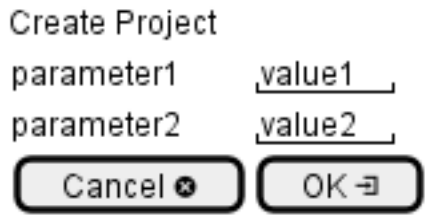
CLI

This is the command line interface for the Create Project Scenario.

```
# caade project create <parameters>
# caade project create exmaple
```

Web Interface

This is a mock up of the Web Interface for the Create Project Scenario.



REST

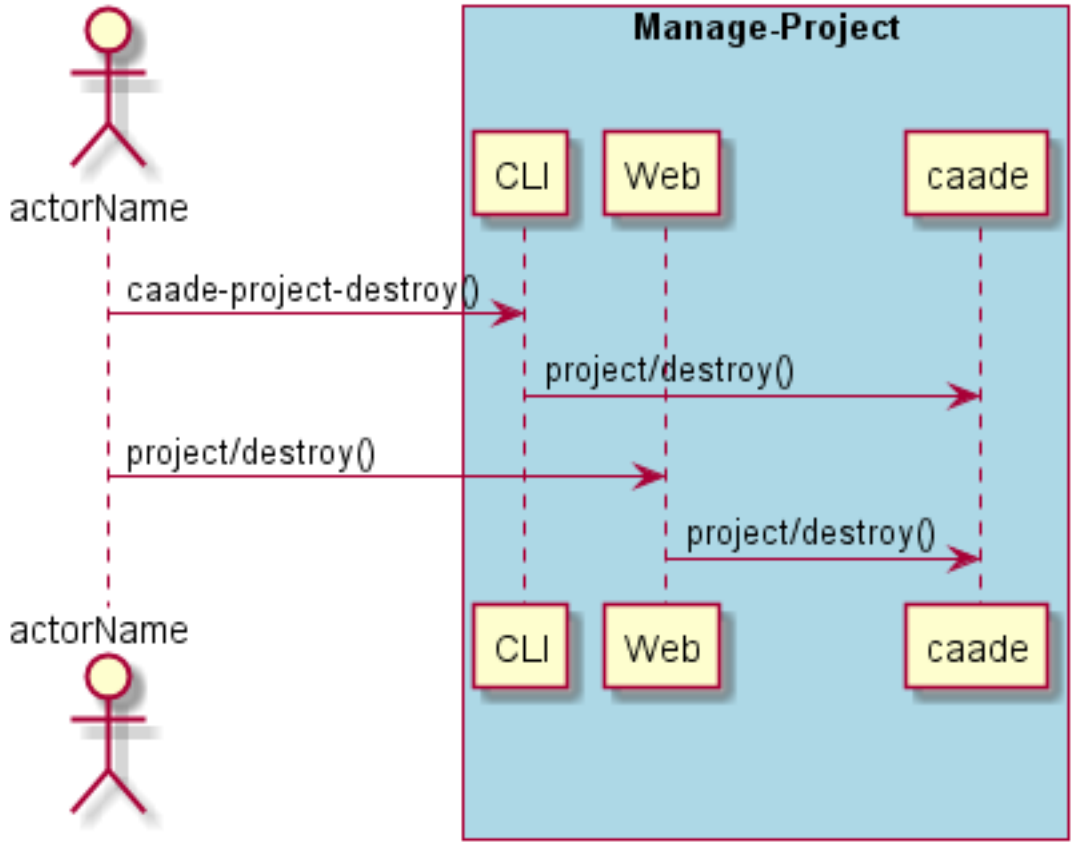
This is the RESTful interface for the scenario.

project/create

Name	Value	Description
parameter1	value1	Description1

Destroy Project

Destroy Project using CLI and Web Interface with ... <parameters>



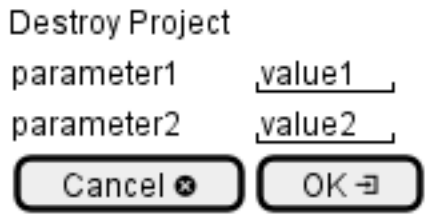
CLI

This is the command line interface for the Destroy Project Scenario.

```
# caade project destroy <parameters>
# caade project destroy exmaple
```

Web Interface

This is a mock up of the Web Interface for the Destroy Project Scenario.



REST

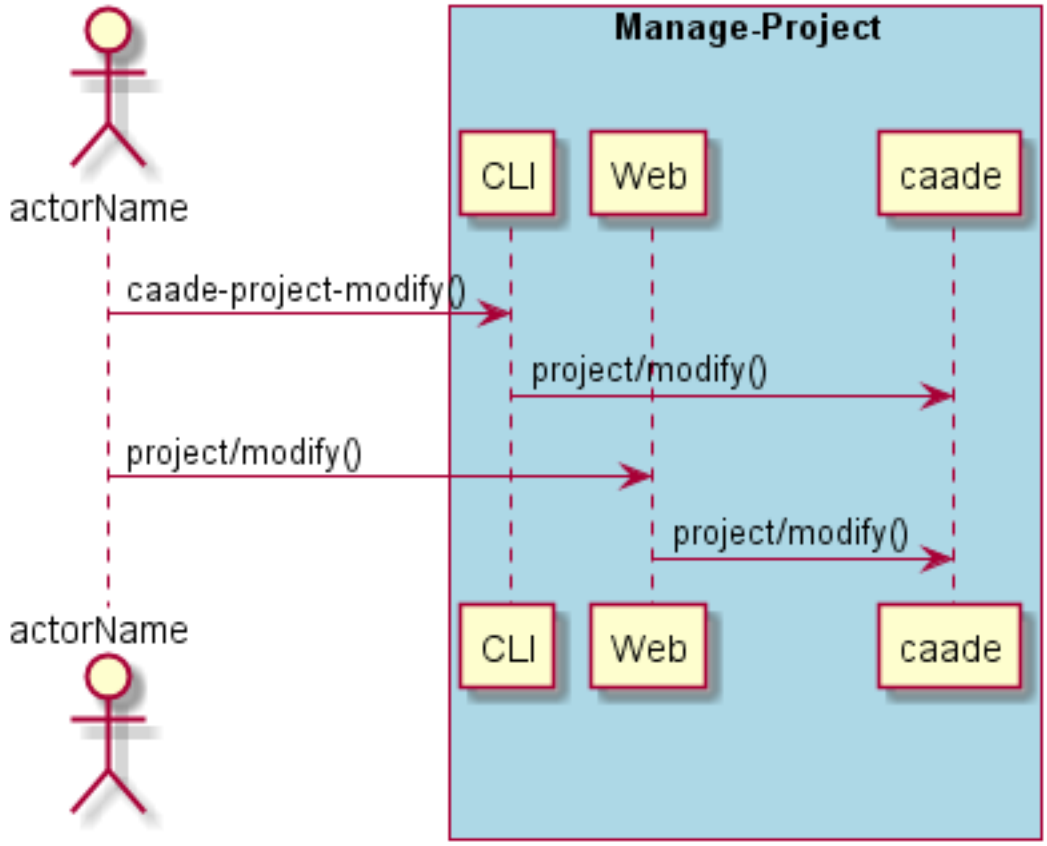
This is the RESTful interface for the scenario.

project/destroy

Name	Value	Description
parameter1	value1	Description1

Modify Project

Modify Project using CLI and Web Interface with ... <parameters>



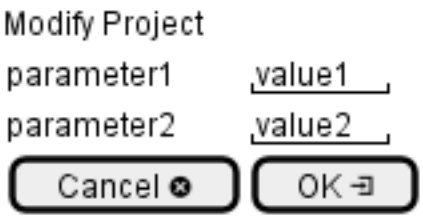
CLI

This is the command line interface for the Modify Project Scenario.

```
# caade project modify <parameters>
# caade project modify exmaple
```

Web Interface

This is a mock up of the Web Interface for the Modify Project Scenario.



REST

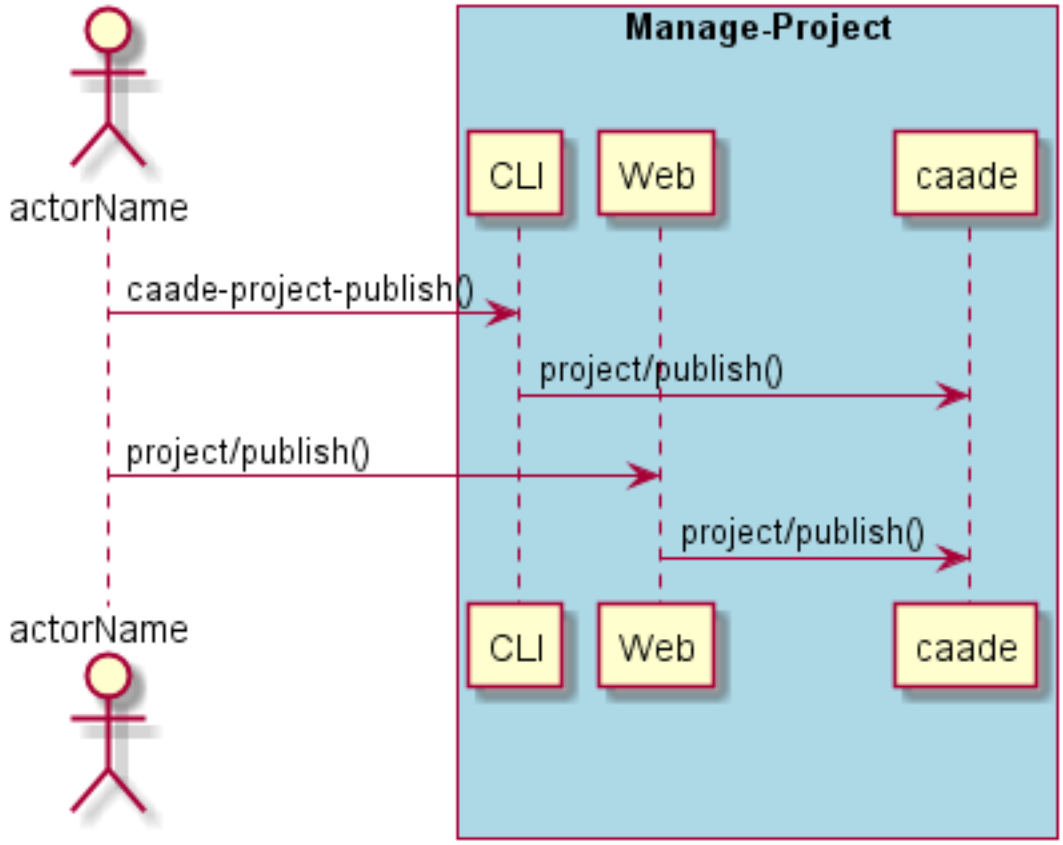
This is the RESTful interface for the scenario.

project/modify

Name	Value	Description
parameter1	value1	Description1

Publish Project

Publish Project using CLI and Web Interface with ... <parameters>



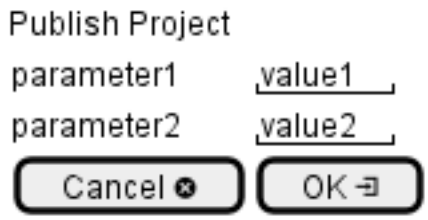
CLI

This is the command line interface for the Publish Project Scenario.

```
# caade project publish <parameters>
# caade project publish exmaple
```

Web Interface

This is a mock up of the Web Interface for the Publish Project Scenario.



REST

This is the RESTful interface for the scenario.

project/publish

Name	Value	Description
parameter1	value1	Description1

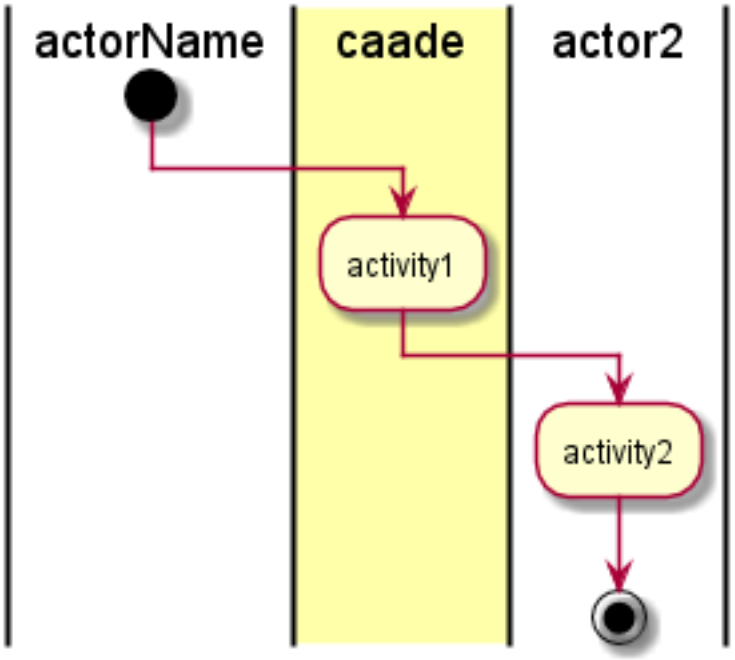
4.7 Manage Service

Add Description

4.7.1 Actors

- *DevOps*

4.7.2 Activities



- Activity from the diagram

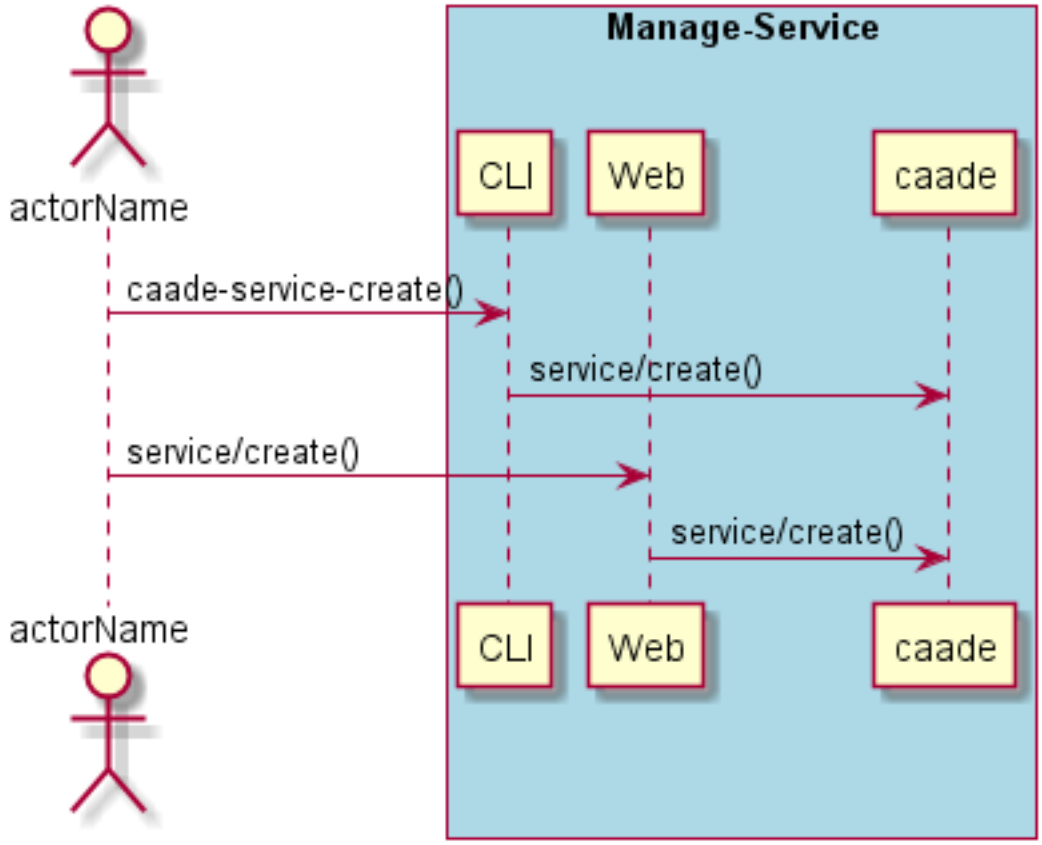
4.7.3 Systems Involved

- *Hybrid Cloud*

4.7.4 Detail Scenarios

Create Service

Create Service using CLI and Web Interface with . . . <parameters>



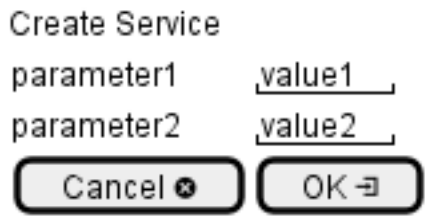
CLI

This is the command line interface for the Create Service Scenario.

```
# caade service create <parameters>
# caade service create exmaple
```

Web Interface

This is a mock up of the Web Interface for the Create Service Scenario.



REST

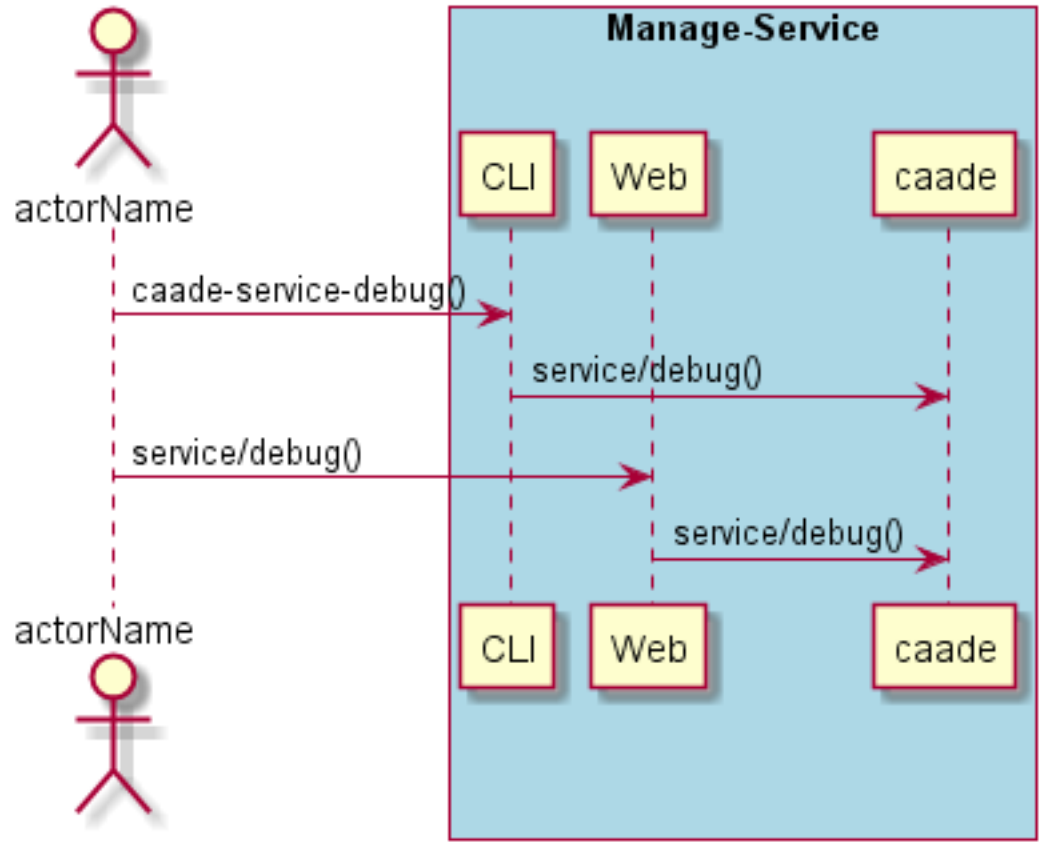
This is the RESTful interface for the scenario.

service/create

Name	Value	Description
parameter1	value1	Description1

Debug Service

Debug Service using CLI and Web Interface with ... <parameters>



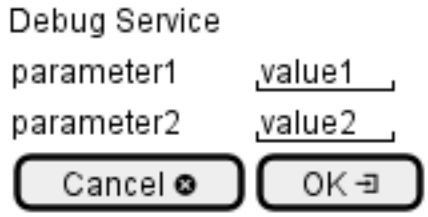
CLI

This is the command line interface for the Debug Service Scenario.

```
# caade service debug <parameters>
# caade service debug exmaple
```

Web Interface

This is a mock up of the Web Interface for the Debug Service Scenario.



REST

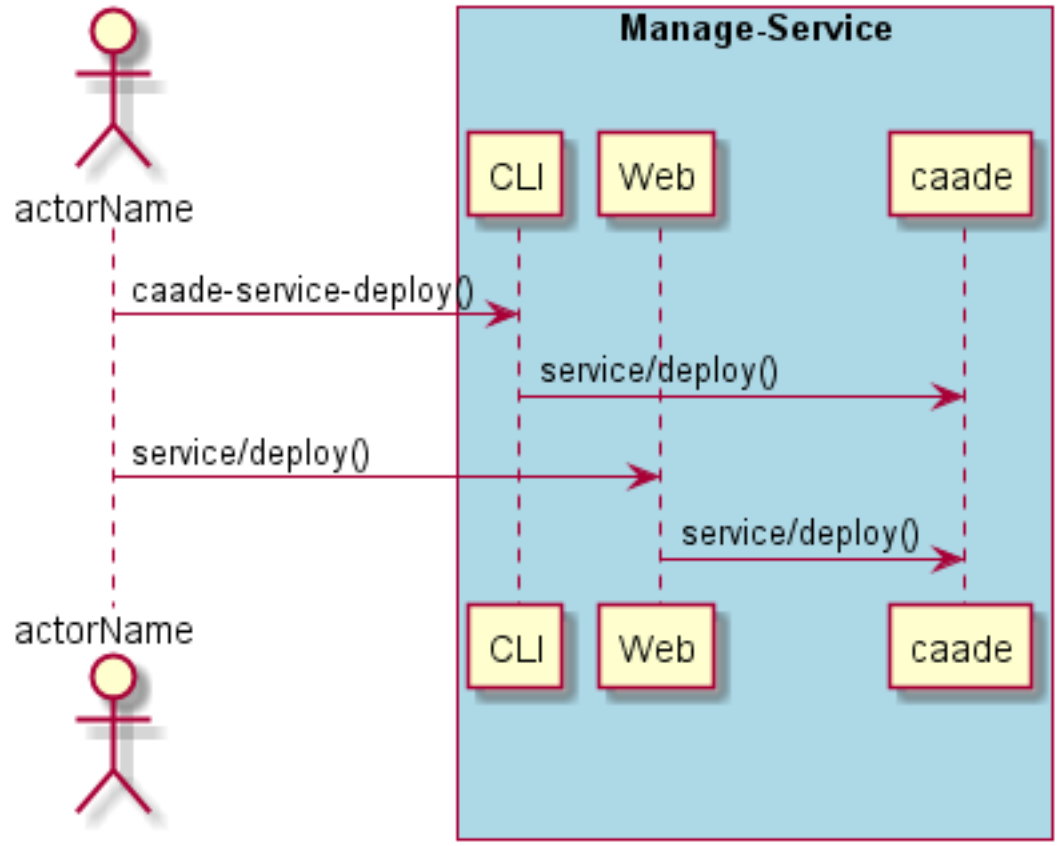
This is the RESTful interface for the scenario.

service/debug

Name	Value	Description
parameter1	value1	Description1

Deploy Service

Deploy Service using CLI and Web Interface with . . . <parameters>



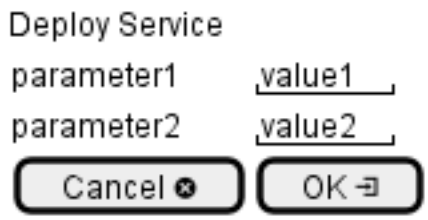
CLI

This is the command line interface for the Deploy Service Scenario.

```
# caade service deploy <parameters>
# caade service deploy exmaple
```

Web Interface

This is a mock up of the Web Interface for the Deploy Service Scenario.



REST

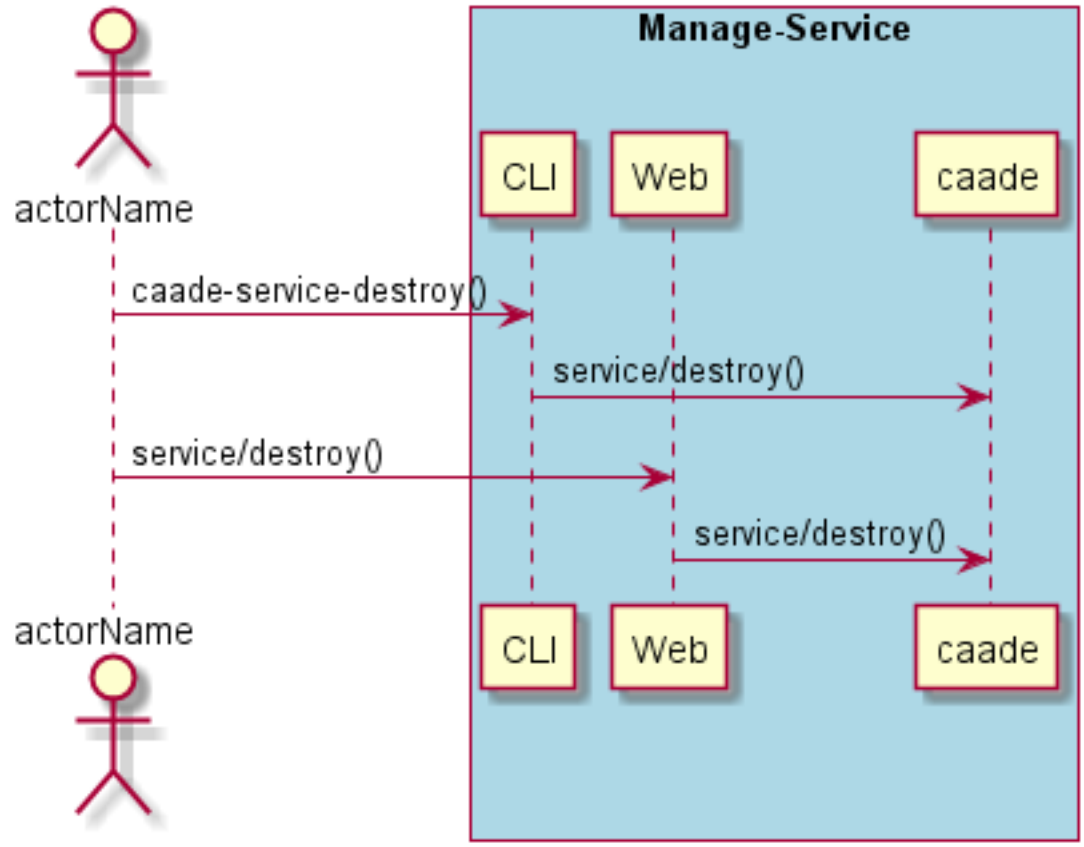
This is the RESTful interface for the scenario.

service/deploy

Name	Value	Description
parameter1	value1	Description1

Destroy Service

Destroy Service using CLI and Web Interface with ... <parameters>



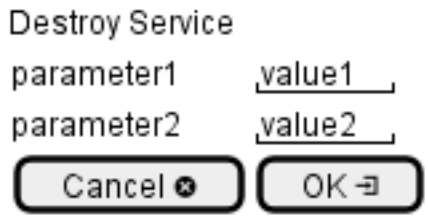
CLI

This is the command line interface for the Destroy Service Scenario.

```
# caade service destroy <parameters>
# caade service destroy exmaple
```

Web Interface

This is a mock up of the Web Interface for the Destroy Service Scenario.



REST

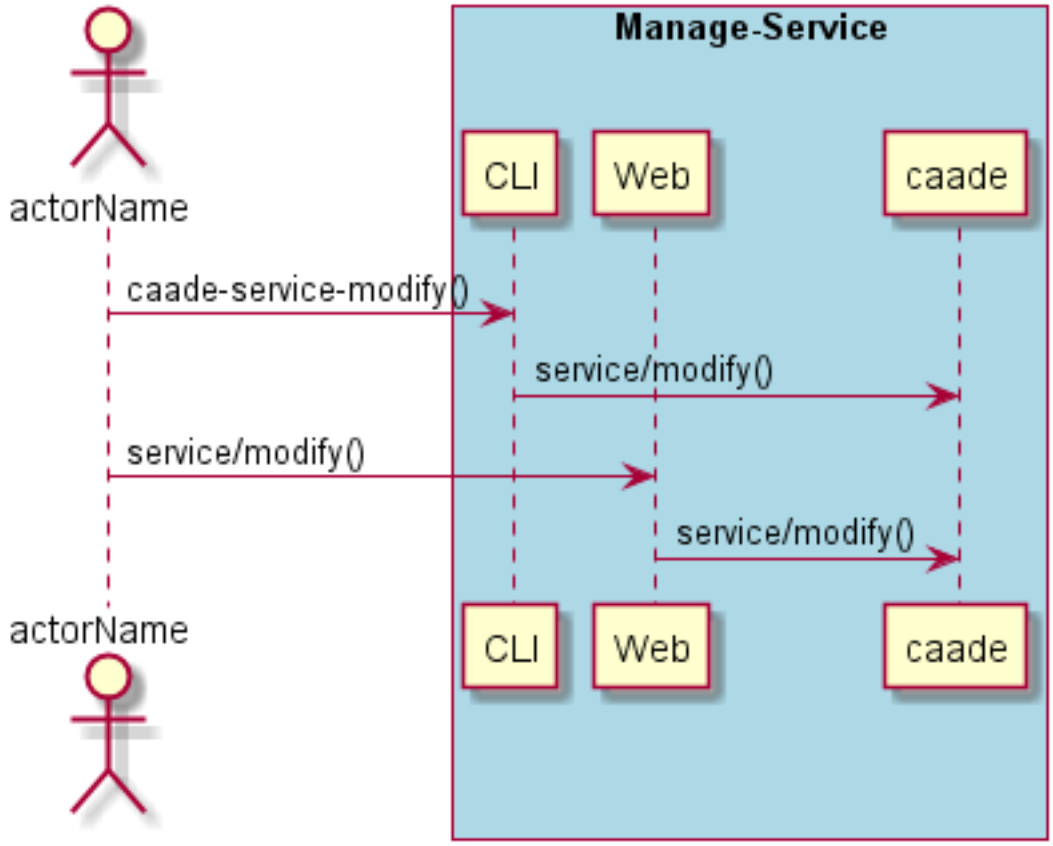
This is the RESTful interface for the scenario.

service/destroy

Name	Value	Description
parameter1	value1	Description1

Modify Service

Modify Service using CLI and Web Interface with . . . <parameters>



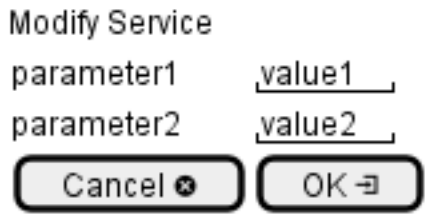
CLI

This is the command line interface for the Modify Service Scenario.

```
# caade service modify <parameters>
# caade service modify exmaple
```

Web Interface

This is a mock up of the Web Interface for the Modify Service Scenario.



REST

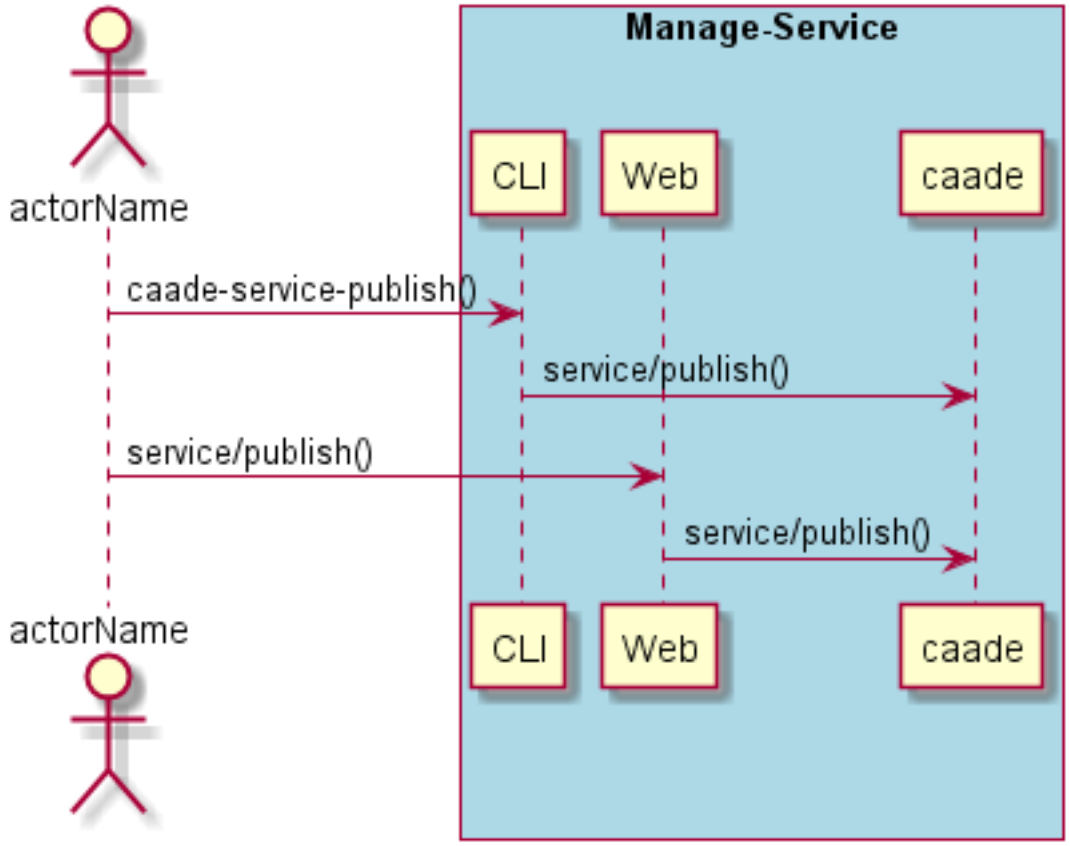
This is the RESTful interface for the scenario.

service/modify

Name	Value	Description
parameter1	value1	Description1

Publish Service

Publish Service using CLI and Web Interface with ... <parameters>



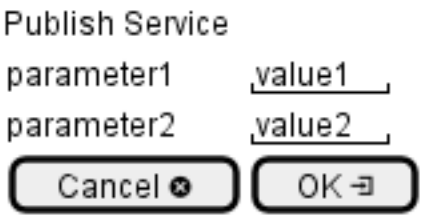
CLI

This is the command line interface for the Publish Service Scenario.

```
# caade service publish <parameters>
# caade service publish exmaple
```

Web Interface

This is a mock up of the Web Interface for the Publish Service Scenario.

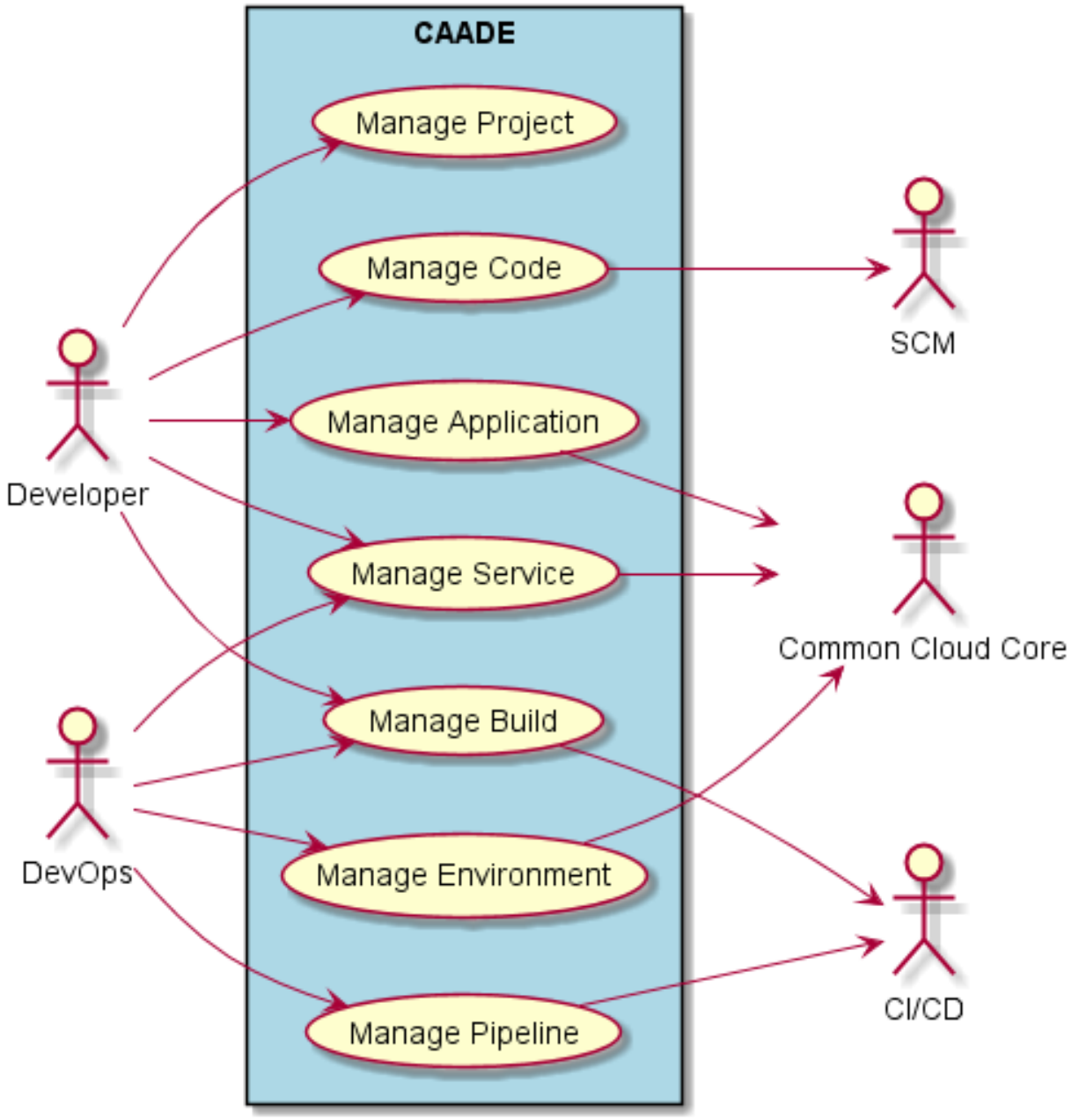


REST

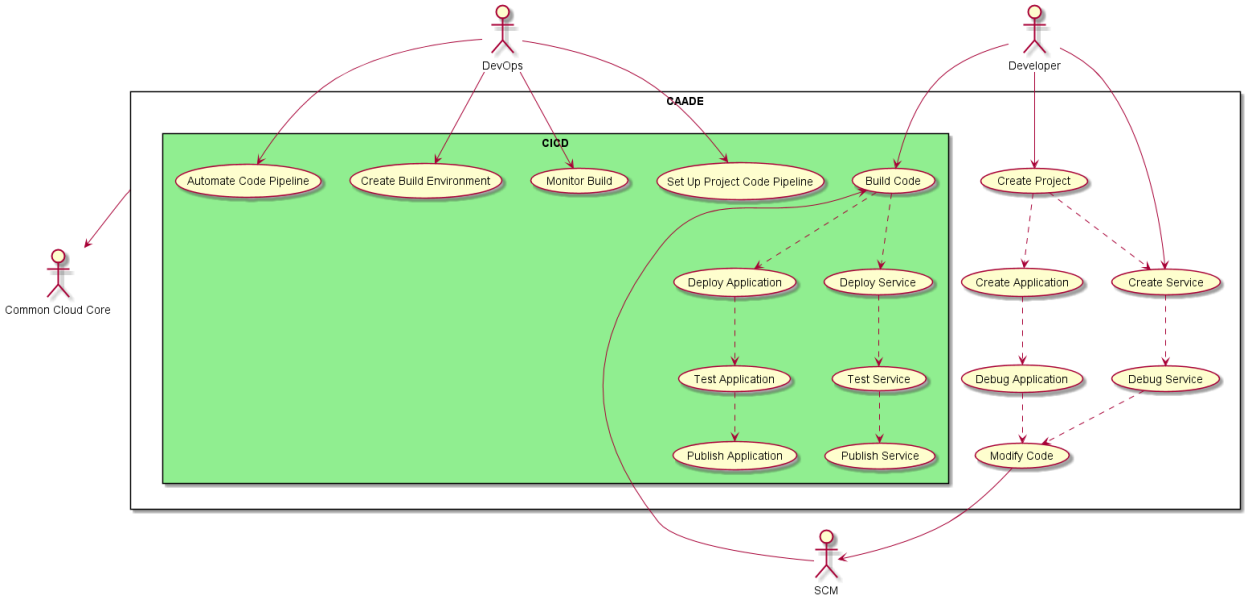
This is the RESTful interface for the scenario.

service/publish

Name	Value	Description
parameter1	value1	Description1



The following is a detailed diagram of the Use Cases and how the scenarios interact with each other.


DevOps
Developer
CAADE
CICD
Automate Code Pipeline
Create Build Environment
Monitor Build
Set Up Project Code Pipeline
Build Code
Create Project
Deploy Application
Deploy Service
Create Application
Create Service
Test Application
Test Service
Debug Application
Debug Service
Publish Application
Publish Service
Modify Code
Common Cloud Core
SCM