Bazooka Documentation

Release 0.3

Bazooka

Sep 12, 2017

Contents:

This is the documentation site for the Bazooka deploy system. If you were searching for the main site go here while the source can be found here

CHAPTER 1

What is Bazooka?

Bazooka is an **Application Release Automation (ARA)** Tool, which helps you to deploy your applications in an automated, repeatable and reliable way managing all the error-prone steps usually done by a human.

CHAPTER 2

Why Bazooka?

Manually deploying an application is a complex and error-prone multi-step process:

1. Getting all your application source files
2. Building the application with your build system
3. Copying the build output to the desired server/network share
4. Modifying the configuration files for the enviroment, changing connection strings and other parameters,
5. Executing all other needed steps such as configuring site options, updating the database, ...

As you can see there are multiple steps and each one is subject to error:

1. You may forget to update your local source code copy with the latest changes from your VCS or you have locally some files which were not checked in, so the application you're building is different from the one everyone else was testing
2. You may have subtle differences in your local configuration like a different compiler or may forget to run a part of the build, like javascript minification and concatenation
3. You may not have access to the folder used for publishing or the one who has is currently on vacation
4. Changing the configuration files manually for each enviroment has some risks like forgetting to change a connection string that now points to a database in another enviroment, or worse you don't remember all the modifications to the configuration so you never touch the configs
5. You may forget to update the database with the new table or stored procedure you have just created or forget to restart the web server.

While points **3**, **4** and **5** may seem severe as your deployment is stopped or your application is throwing exceptions due to wrong application or system configurations point **1** and **2** are the real problem as the deployed application is now a snowflake that cannot be recreated for testing or rollback in case of emergency.

As you can easily see each of these steps introduce complexity in your deployment process and a risk of making an error, especially when tired or under pressure.

Now multiply these steps for maybe three or four enviroments (Test, UAT, Staging and Production) and for ten to twenty applications (not too many, for a mid-size business) and you have a problem on your hands not to mention all the time subtracted to other activities.

The only solution to this problem is to completely automate your application deployment, with the objective of being able to go from a version in your VCS to deploying your application without any manual intervention.

Bazooka was created for this precise situation, to be able to package an application and deploy it reliably and automatically across different enviroments without any manual intervention.

CHAPTER 3

How does it work?

The idea behind how Bazooka works is really simple:

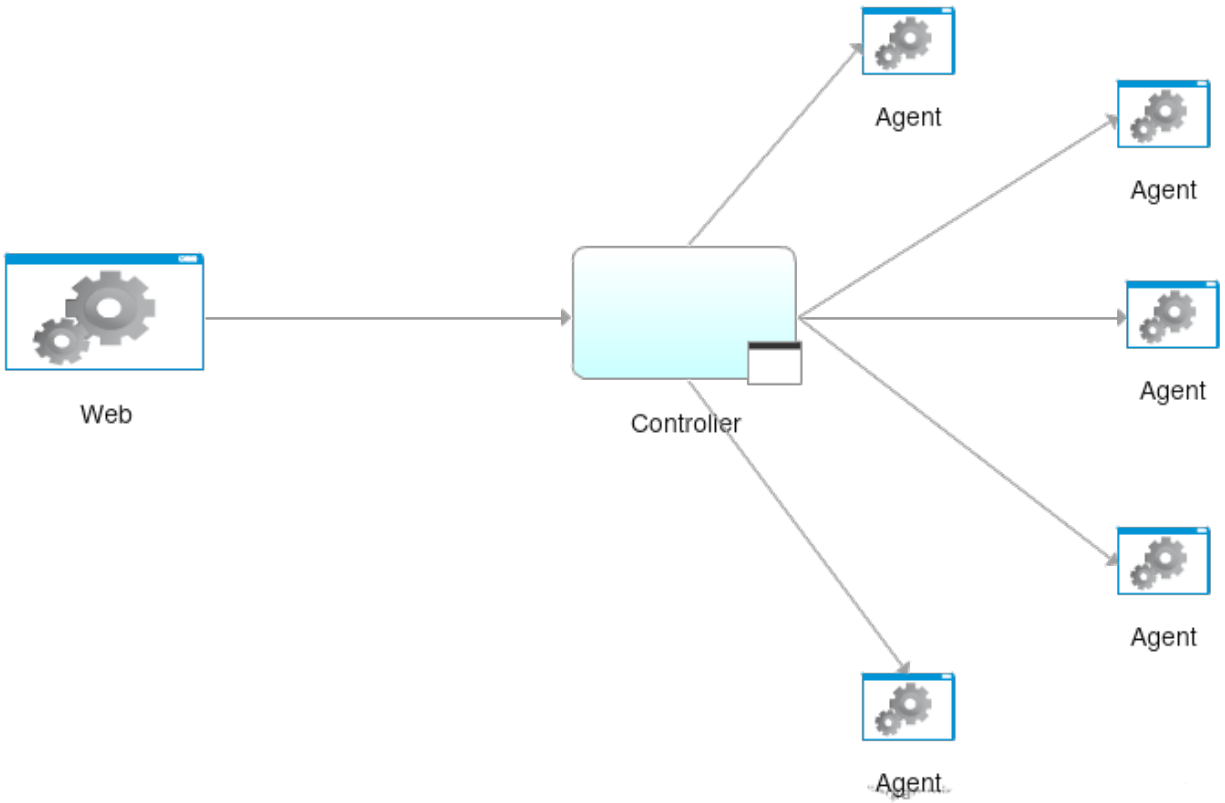
1. Package your application after it has been built by your build system in a format that allows versioning (as it turns out NuGet packages work very well for this)
2. When you want to deploy your application select the version you want to publish and let Bazooka deploy it according to the configured steps

Installation

System overview

Bazooka is a web application composed ot three main parts:

- a **Web site** which will allow you configure the system, set permissions and deploy applications
- a **Controller** which is a windows service dedicated to running tasks like sccheduled deployments, cache cleanups, periodic health checks and log compactions
- one or more **Agents** to be installed on the machines where you will deploy your applications. They will execute all the commands from the controller allowing deploys to be executed



To install Bazooka you must first download the latest release from Github and then we can start.

Database

The first thing to install will be the database. SQL Server 2008 ad 2012 have been extensively tested but newer versions should be fine nonetheless.

To install the database, simply create a new Database in you SQL Server instance and then apply the *dacpac* file inside the **database** package downloaded from Github (If you don't know how to apply a dacpac file a guide can be found here) .

Controller

The Controller is a console application based on Topshelf that can be installed as a Windows service. Running the controller as a caonsole application is only useful to debug configuration errors and it is strongly recommended to run it in service mode.

To install the Controller simply unpack the Controller archive downloaded previously and copy its content to a folder on the machine where you want to run the controller. After that the servvice can be installed as an Administrator from the command line with

Controller.exe install

For more options you can consult the Topshelf docs .

After installation you can proceed to edit the configuration file to change the connection string to the previously created database then start the Controller service.

Note: Make sure that the account running the service can connect to SQL server and the database. Your system administrator may want to create a dedicated user to tailor its permissions

Web site

The Web site is a common MVC Application so to install it you will first need to configure an IIS Web Site to install it. After having configured the website you can simply unzip the content of the **Web** archive in the publish directory and proceed to modify the Web.config file to suit your needs.

The most important parameters to configure are:

DataContext The connection string to the database. Change it to point where you created the database

activeDirectory Activates active directory authentication instead of the standard form authentication.

adDomain If you choose to authenticate with active directory this must be set to the Active Directory Domain you users will belong to

Agent

The last part of the system is the **Agent** which will have to be installed on every machine where you will deploy your apps. Its role is simple: it will listen for commands from the controller and execute them to complete the deploys.

Note: You may want first to discuss with your system administrator under what account the Agent will run. As it will need to execute scrips and access specific directories he may want to taylor the necessari permission with dedicated accounts

To install an Agent on a machine the first thing to do is reserve the 9000 port which it will use to listen for commands from the controller. This can be done with the netsh command.

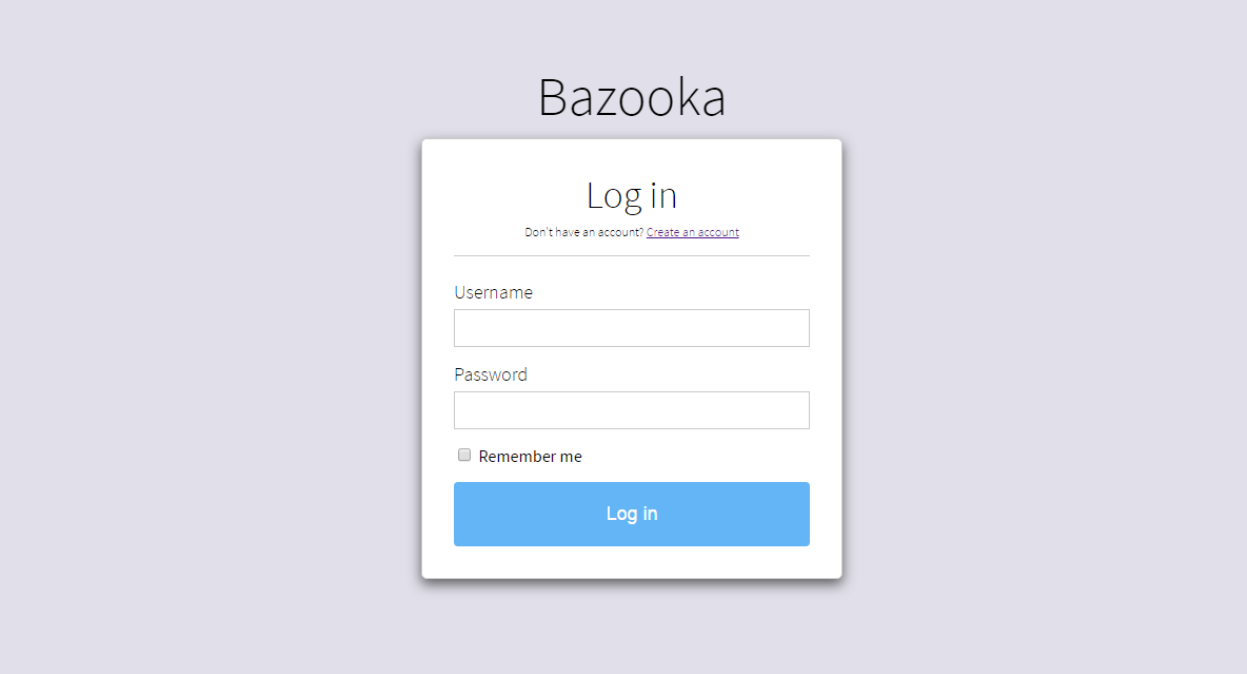
Once reserved the port the service can be installed in the same way the **Controller** was installed

Configuration

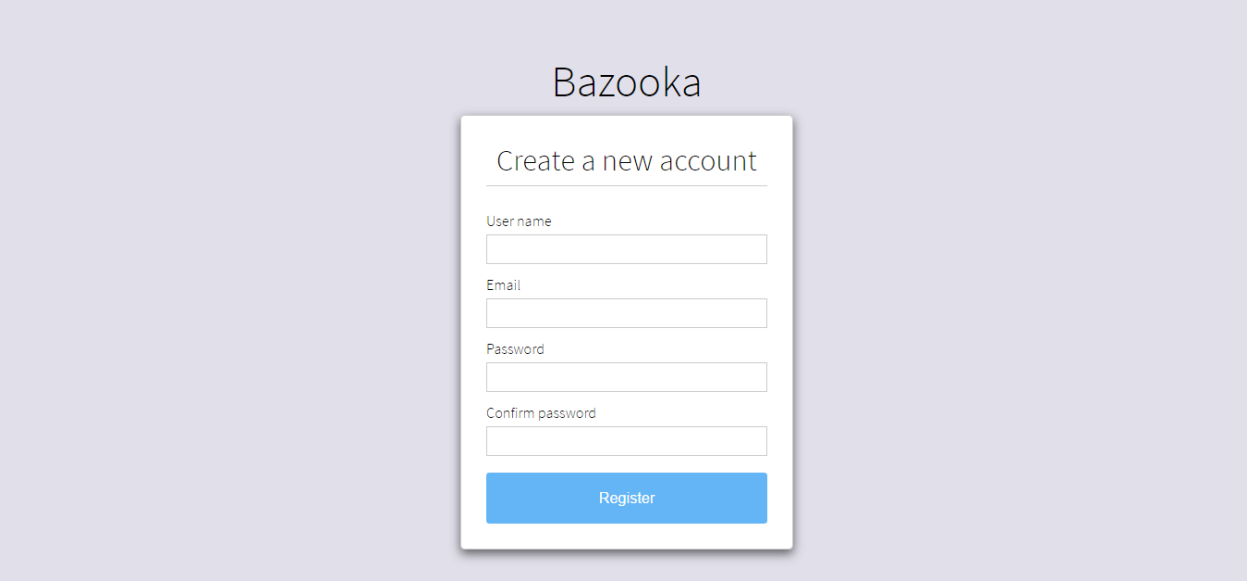
This section assumes you've already installed Bazooka and packaged your applciation and will guide trough the necessary steps to configure and deploy your first application.

Creating your first user

The first thing to do is to login into Bazooka. Point your browser at the URL assigned during website creation and you will reach the login screen.



The first thing to do is to create your first user by clicking the **Create an account** link. You will see the account creation page where you can input the new user data.

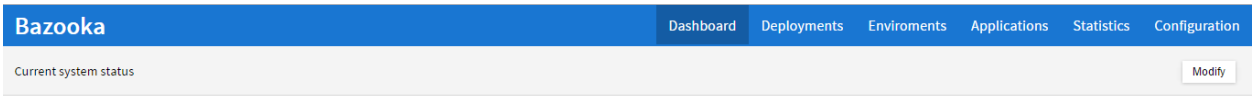


Note: if you have chosen to use Active Directory authentication use the same password to create the user as it will be validated against your Active Directory Controller

Note: The first user created in the system will be automatically an Administrator with full privileges. This can be changed later if it is not adequate.

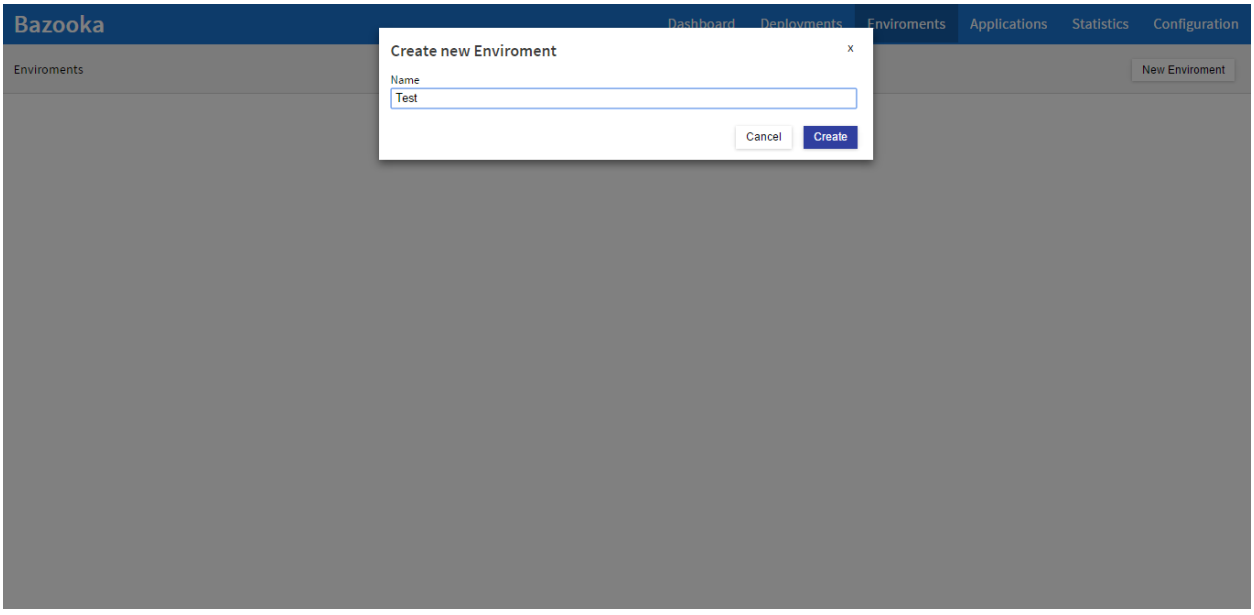
Taking a look around

Once logged in you will reach the Dashboard, Bazooka's main screen. As of now no application has been configured so it is empty but we will soon change this.



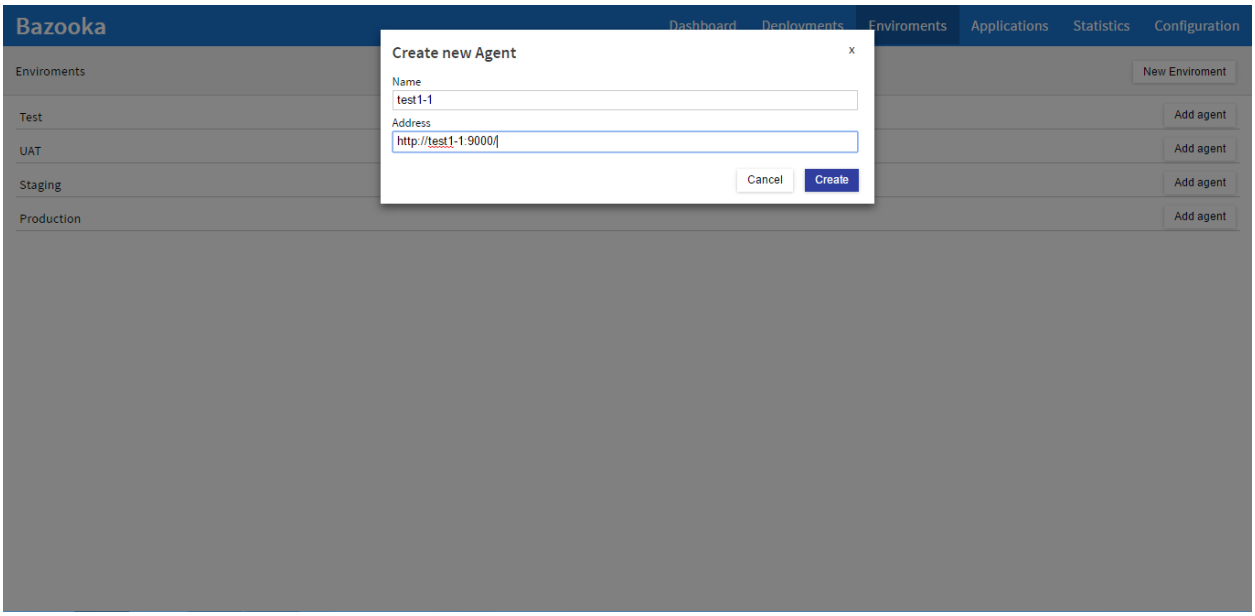
Creating your enviroments

The next step is enviroments definition and the addition of at least one agent per enviroment. Go to the **Enviroments** section by clicking on the section in the header with the same name. In this page you canc click on the **Create new enviroment** button to create your enviroments. A common organization is Test- UAT (User Acceptance Testing) - Staging and production.



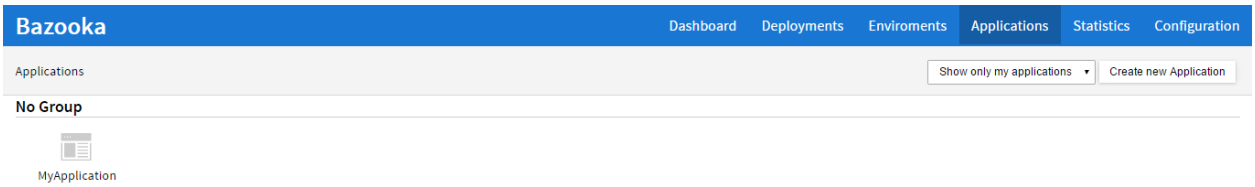
Adding an agent

Once all the enviroments have been created you can add agents to any enviroment by clicking on the **Add new Agent** button.

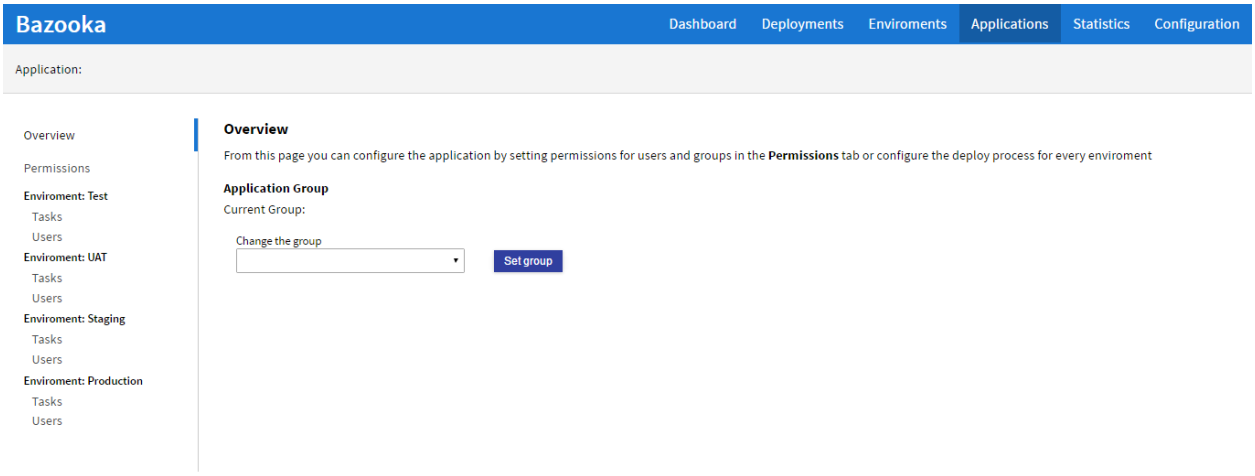


Configuring your application

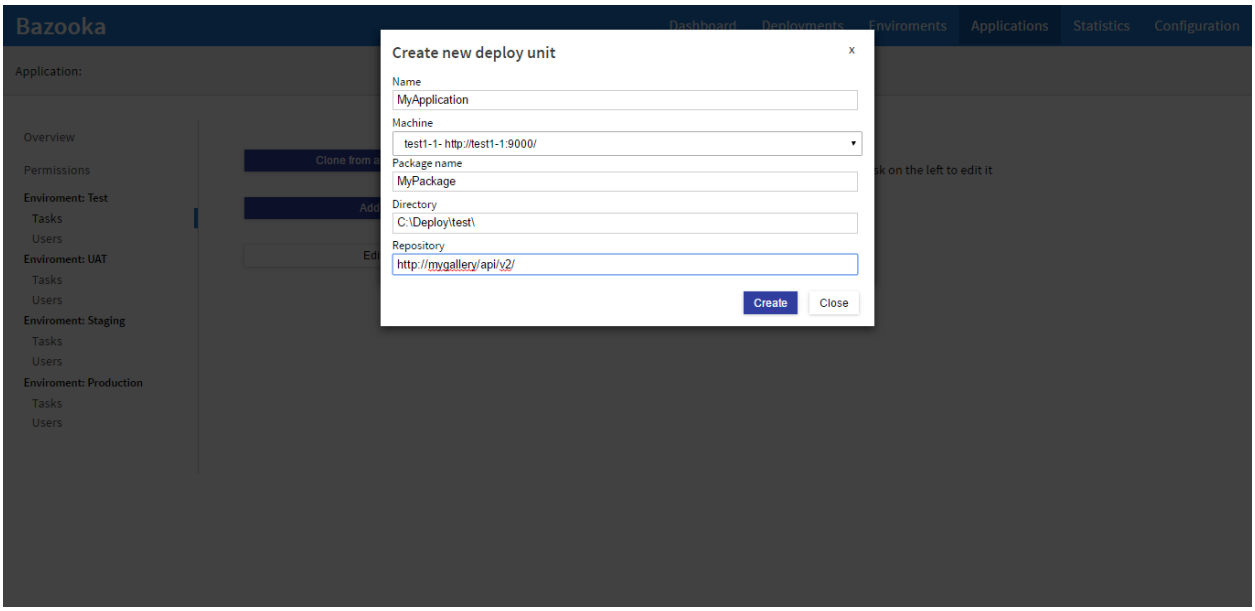
Once the enviroments and agents have been configured we can proceed to configure our first application. To do this click on the **Applications** tab in the header and then click on the **Create new Application** button. After having chosen a name we can see it and by clicking on it edit its configuration.



Here on the left we can see, for every enviroment, a **Tasks** tab which will allow us to configure the deploy process and a Users tab whic will configure the users or groups of users allowed to deploy the application.



Next, click on the **Tasks** tab for an enviroment. Here you can see the list of task that comprise your deploy. Click on the **Create new task** button and choose the **Deploy task**.



Next insert the necessary info for the task :

Name The name of the task, useful to distinguish them in the list

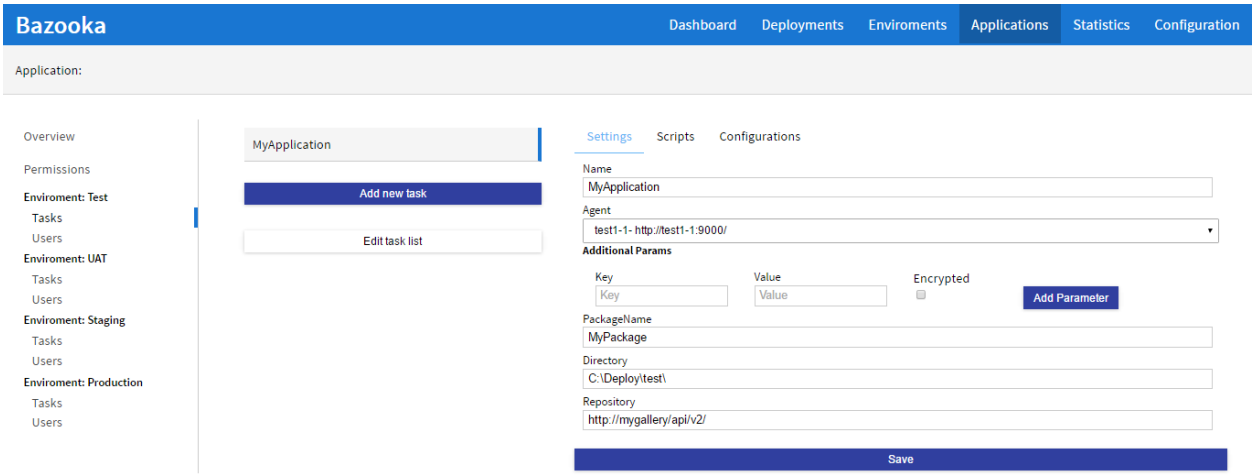
Machine The agent to use for the deploy

Package name The name of the nuget package containing your application

Directory The remote directory where the package will be copied

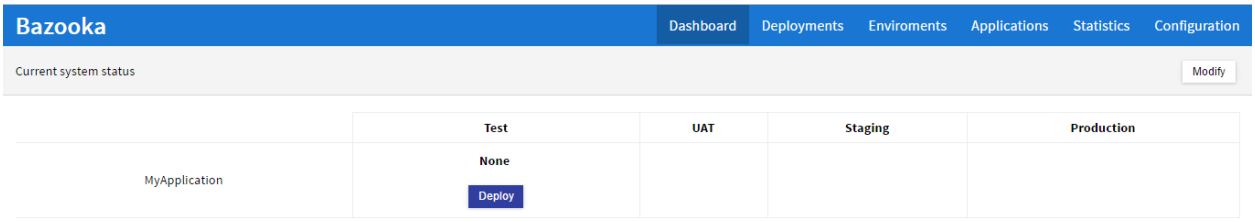
Repository The place where the package can be downloade. It can be a nuget gallery o a simple network folder but must be reachable from the agent

Once done click the **Create** button and you will see the created task on the left. Now we are ready for our first deploy.



Your first deploy

To deploy our application we can simply return to the **Dashboard** by clicking the link in the Header. here we will se our new application and for each enviroment a cell indicating the currently deployed version and a **deploy** button.



By clicking on the deploy button you will see the deploy dialog where you can select the version to deploy. Choose the preferred version and click the deploy button.

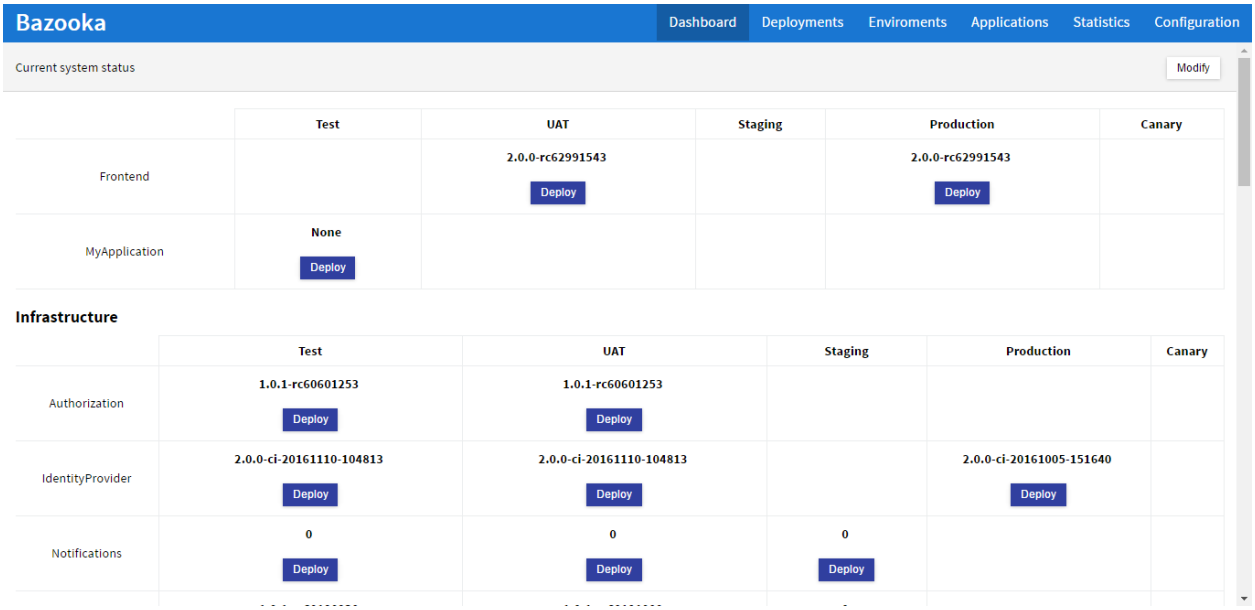
Your deploy is now started. You can go to the **Deployments** tab to see your deploy and click on it to see the logs and its advancement

Walktrough

Bazooka has many different sections in its interface so it's useful to walk trough them one at a time.

Dashboard

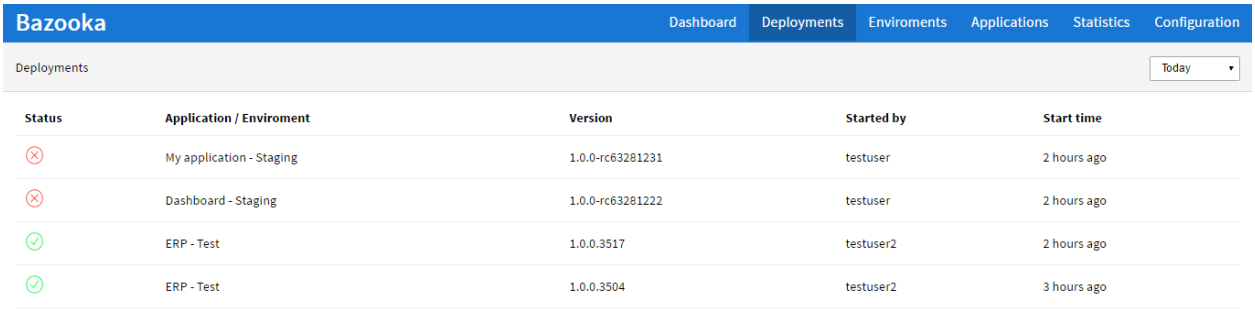
The first section and the one you'‘l see once you login is the **Dashboard**. here you can see at a glance the status of the entire system (or the part of it you are allowed to see). For each application and every enviroment where you have access it is shown the currently deployed version allowing you to quicily deploy a new version.



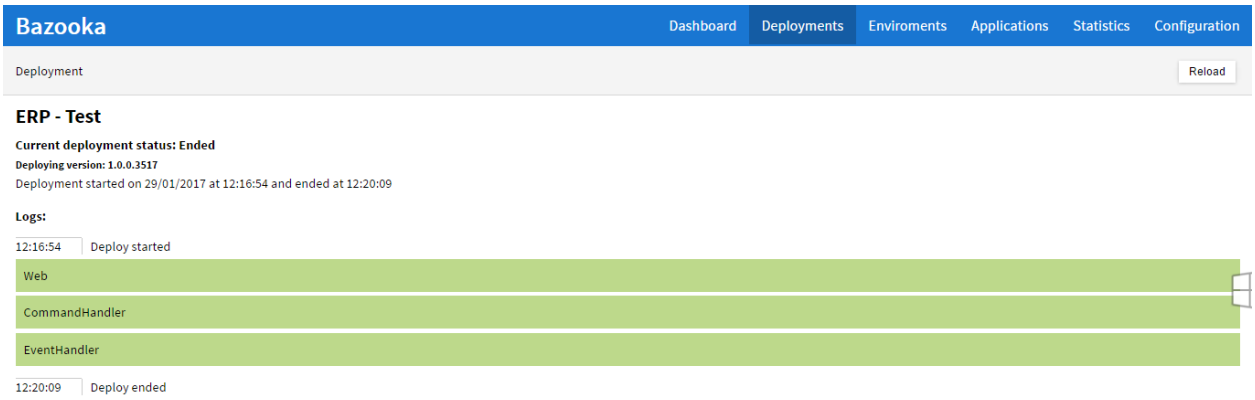
Optionally you can define groups of applications to more easily organize them visually. These groups can be then ordered by pressing the **Modify** button in the top left cornet and then ajdusting the position with the arrow buttons on each application group.

Deployments

The second tab and one of the most used is the **Deployments** tab. In this tab you can see your last deployments. For each deployment the first column indicates its state while on the rest of the line you can see the application / Enviroment where the deploy is happening, the user who launched it and the relative time it started.



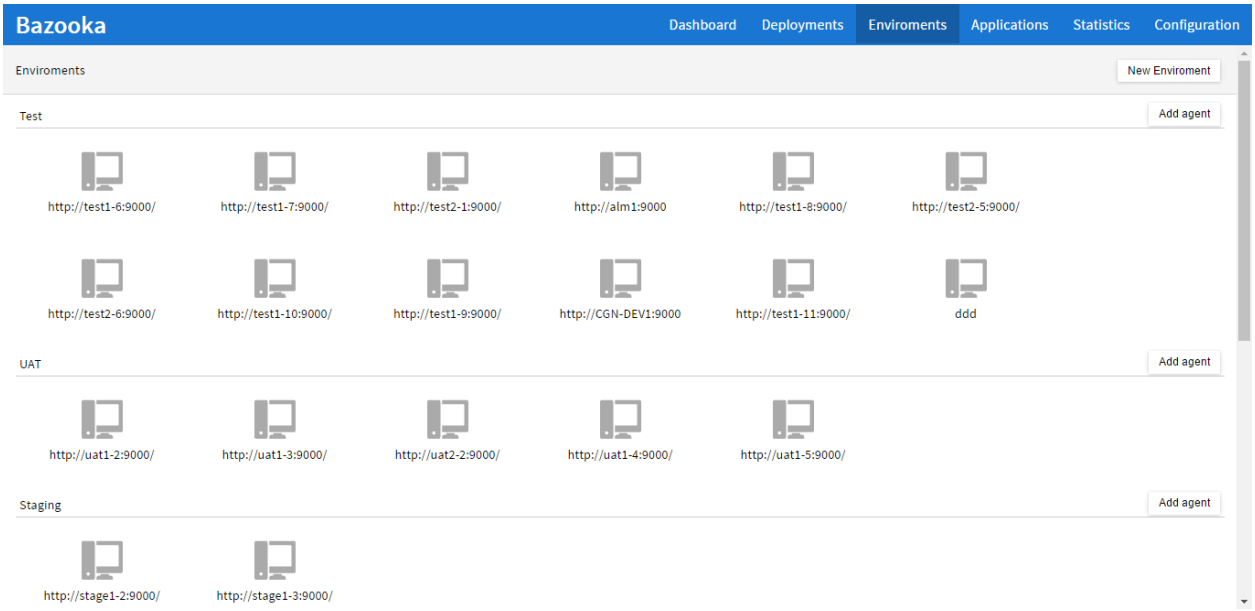
By clicking on a deploy we can see its status in more detail. If the deploy is still running its status will be automatically refreshed every 10 seconds.



On the top left corner a drop-down will allow you to view older deployments.

Enviroments

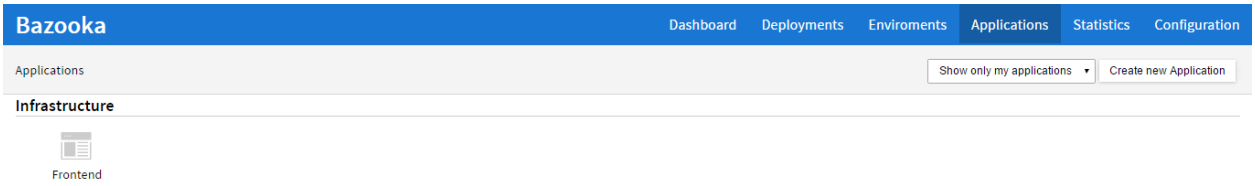
The **Enviroments** section allows you to view and define the enviroments where your applications will be published and, for each enviroment, to register agents as targets for deployments.



In this page you can create new Enviroments by clicking the **Create new enviroment** button in the top left corner. Once one or more enviroments have been created you can start adding agents to each enviroment by clicking the **Add agent** button.

Applications

The **Applications** section allows you at a glance to view all the configured application in your system. Normally you will see only the applications for which you are an administrator or all if you are a global administrator but you can always see them all by using the drop down on the top left corner. If you have defined application groups the apps will be groupd by them allowing for a clearer view.



By clicking on the **Create new application** button you can create a new application (requires administrator privileges).

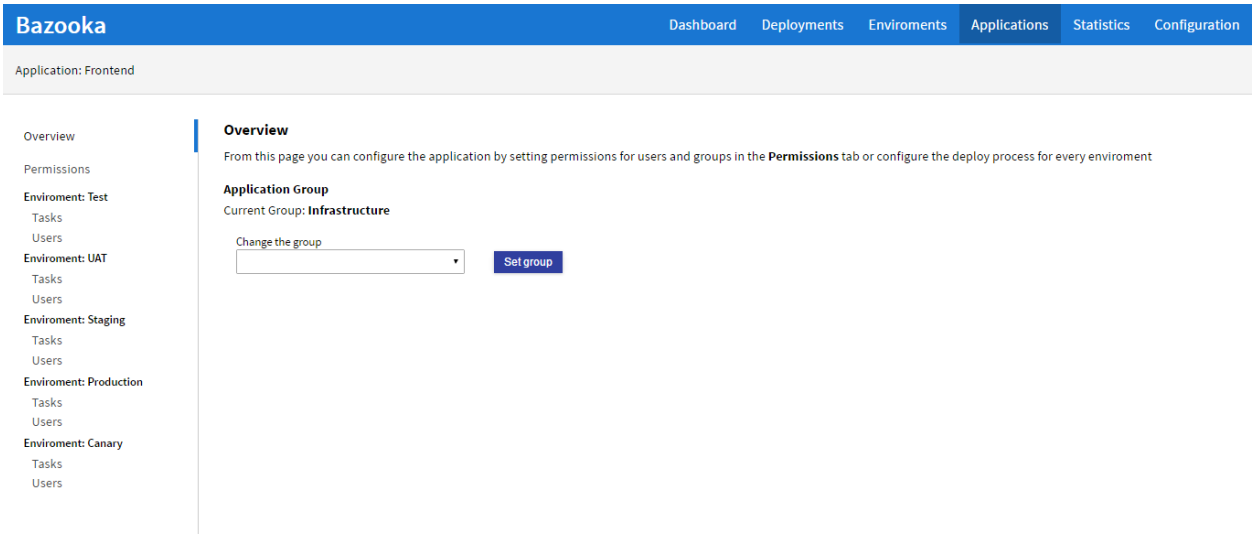
By clicking on an application we can go to its configuration page where we will found a tabbed interface. On the right side of the page we can see some tabs:

Overview gives some general infos about the application and allows you to set the application group

Permissions allows you to define an application administrator, someone with full privileges only for this application

Tasks one tab for each enviroment this will allow you to define your deployment process and its composition in tasks

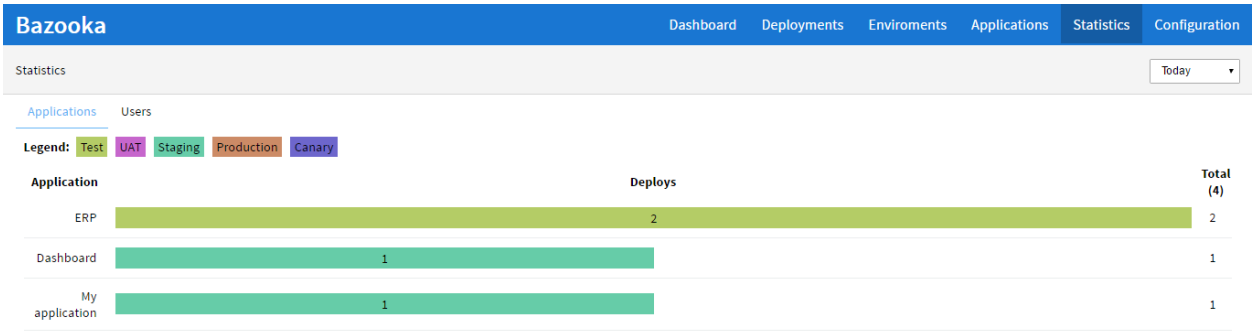
Users one tab for each enviroment allows you to give permissions to publish to specific users or groups



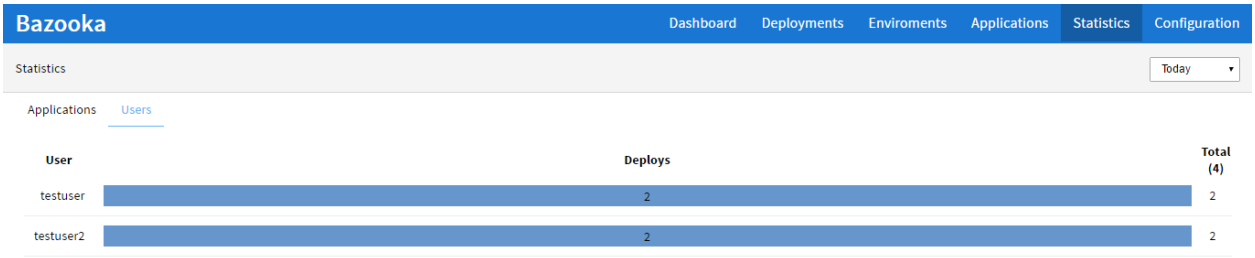
Statistics

The statistics page allows you to see some statistics about the usage of Bazooka in the specified period of time, configurable by the drop down in the top left corner. This section is divided in two tabs: **Applications** and **Users**.

The **Applications** tab will show for each application the number of deployments for each enviroment allowing you to spot irregularities like an application being only deployed to production.



The **Users** tab instead will show for each user then umber of deploys made in the specified period of time.



Configuration

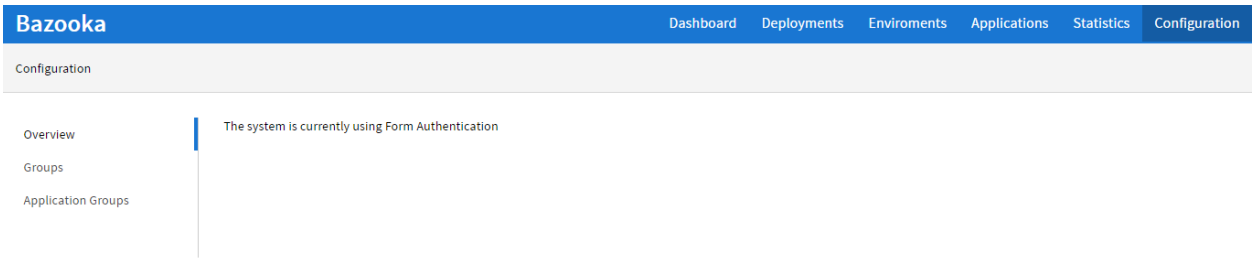
The configuration section contains configuration for the entire system, as opposed to those specific to an application. This page is divided into several tabs:

Overview will give you some informations about the system

Groups will allow you to configure groups of users for easier handling of permissions

Application Groups will allow you to create application groups to group your applications

Task templates will allow you to create, configure and modify task templates, a skeleton of a task where you can specify parameters to fill when used



Packaging

Bazooka works on a simple idea: if an application can be packaged with all his dependencies and configurations and versioned then that package can be used to deploy the application in various enviroments in a repeatable way.

Packaging format

To package the applications we had to choose a format. Ideally it should be:

- **versionable** as every package should be easily marked with a version number
- **easy to create and modify** with existing tools as you probably alreaady have a build or CI system and it should be easily to produce the packages from it
- **compact** as it will be necessary to store a good number of versions of the applications and we can't afford to waste too much space
- **easy to manipulate** with existing tool and libraries, as having prebuilt tools avoids having to create an entire ecosystem from scratch

The choice, given these principles, was easy: Nuget.

NuGet born as a format to distribute compiled dlls for the .NET ecosystem allows to package a set of files in a compressed archive and attach some metadata like version, owner, a link to the release info page. With the format have also been created a multitude of libraries and tools to create, manipulate and host Nuget packages allowing us to reuse all that work.

This choiche has been also made by many others with similar requirements like Chocolatey.

Packaging your application

A package should contain everything needed to deploy you application which means:

- all compiled binaries
- all static assets (css, javascript, view files)
- optionally an installation and uninstallation Powershell script (these can be included in the package or configured in the web application)
- configuration files and optionally config transforms for any enviroment (these can be included in the package or configured in the web application)

Creating a package

Creating a package is really straightforward and can be easily automated and included in an automated build. For each package to create (an application may consist of multiple packages, one for the web site, one for a windows service, ...) it's necessary to create a definition file, the **nuspec** file.

A **nuspec** file is simply an xml file with the nuspec extension that tells Nuget how to create the package. To begin you can simply take this as an example and modify it for your project:

```
<package>
  <metadata>
    <id>MyAwesomeWebSite</id>
    <version>$version$</version>
    <owners>Awesome Inc</owners>
```

```
    <description>My Awesome Web Site</description>
  </metadata>
  <files>
    <file src="PATH\TO\FOLDER\**\*.*" target="" />
  </files>
</package>
```

To adapt this example to your specific case you have to modify:

id tag change the content of this tag to the name of the application contained in the package (no spaces or characters not allowed in URLS, this is a nuget convention)

version tag the $version$ syntax means that this is a placeholder that will be replaced when building. Better leave this here and specify it later

owners tag Specifies the owner of this package. It's only a description and can be omitted.

description tag A description of the package. It's only a description and can be omitted.

files tag This tag and all it's children specify what files will be added to the package. You can replacce PATHTOFOLDER with the path where all the applications file can be found but leave the trailing ***.* as that means that all the files will be recursively added.

Once you have created the nuspec file to create the package you just have to call the Nuget executable in this way

nuget.exe pack YOURNUSPECFILE.nuspec -Properties "version=YOURPACKAGEVERSION"

replacing **YOURNUSPECFILE.nuspec** with the actual name of your nuspec file and **YOURPACKAGEVERSION** with the version of the package you want to build. This command will create a file with the nupkg extension in the current folder that contains all the applications file. If you want to inspect it you can simply change the extension to .zip and open it like a normal compressed archive.

Hosting your packages

Once you have packaged your application in a nuget package it is necessary to host them in a **repository**. A repository can be created in many ways

- you can simply use a shared folder even a network folder
- you can host a Nuget Gallery on your own server.
- you can use one of the services offering to host your NuGet feed

Deploy

Bazooka allows you to define your application deploy process by composing different tasks. Each task can be of a different type and have different options to allow you to customize your deploy.

Deploy task

The Deploy task is the most common task used in Bazooka. It will allow you to copy a nuget package in the specified directory to the specified machine (Agent).

This task will

- copy the package in the specified golder on the target machine

- extract its contents
- search and apply an enviroment specific transform if present. (For example if we're deploying to the Test enviroment and a web.Test.config file is present il will be considered and XDT transform and applied to the web.config file)
- if a file named Install.ps1 is found it will be executed to complete the install

Note that if a package is already present it will first proceed to

- execute a Uninstall.ps1 script if found
- delete the existing package contents

before attempting to install the new version.

Mail task

The mail task is a task designed to send a mail to one or more recipients to warn them of a deploy. This task allows you to specify a mail Text, a sender, and one or more recipients separated by a semicolon.

Local script task

The local script task is simply a way to make the Controller xecute a Powershell script. the only parameters are the name of the task and the script to execute.

Remote script task

The remote script task is a variant of the local script task. In this case the script will be executed by a specific agent.

Database task

The last type of task is the **Database** task. This task will allow you to deploy a **dacpac** file contained in a package to a specified database. Once this task executes the specified package will be fetched, its contents will be searchd for a dacpac file then it will be applied to the specified database aligning its structure to the one specified in the dacpac file.

This task accepts the following parameters:

Name The name of the task

ConnectionString The connection string to the database to the deploy

Package The name of the package containing your dacpac file

DatabaseName The name of the database to deploy

Agent The agent which will execute the deploy. Note that this agent must have suitable permissions to access and modify the database

Templated task

The most recent type of task added. By configuring a template for your task in the **Configuration** section or using one of the already defined ones you can specify the needed parameters to execute a task based on a task template.

You will have to specify the following parameters:

Name The name of the task

Task The task template to use

Package The name of the package containing your application

Repository The repository where the package can be found

Agent The agent which will execute the deploy.

Along with all the parameters specified by the task template