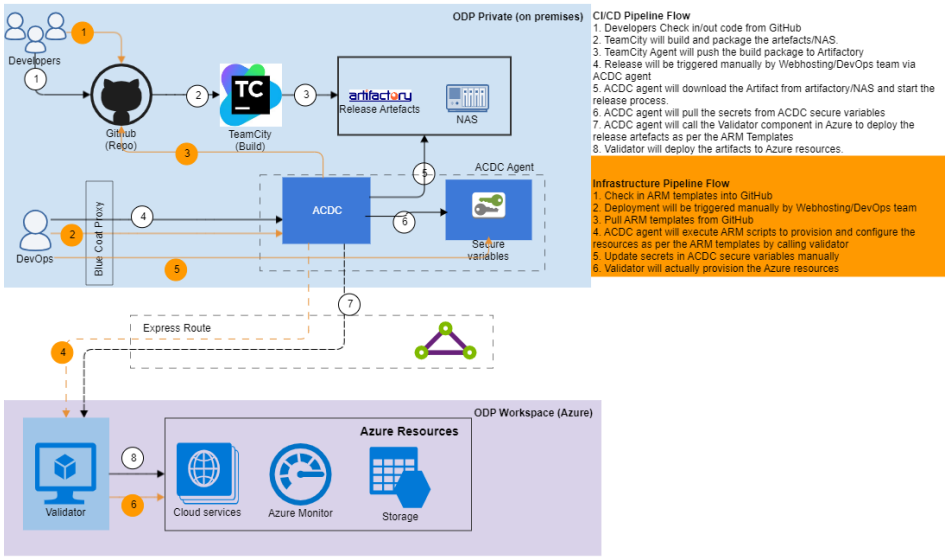
Implementation Overview

As part of the migration of FirstNet web tier applications from ODP IF to One Cloud Public, the deployment pipeline will be migrated from current ACDC tool. The new pipeline will be set up in Octopus deploy.

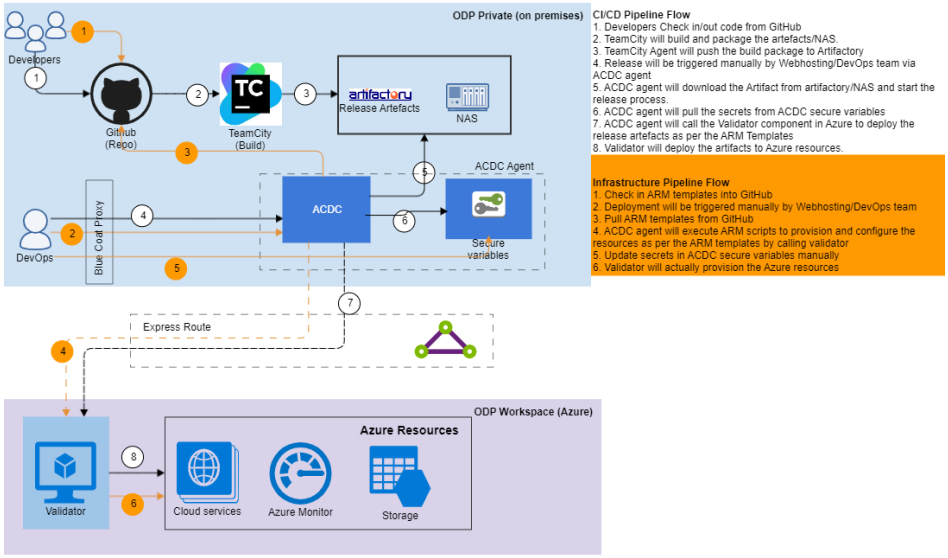
Configuration files

Configuration files and secrets will be migrated from ACDC to Octopus deploy. We will use powershell scripts in Octopus to fetch the secrets and configurations from ACDC tool. These configurations and secrets will be pulled *as is* and will not be modified so that they can be referenced directly from within the source code. The format for the configuration key name will be : *SystemName.ComponentName.KeyName*

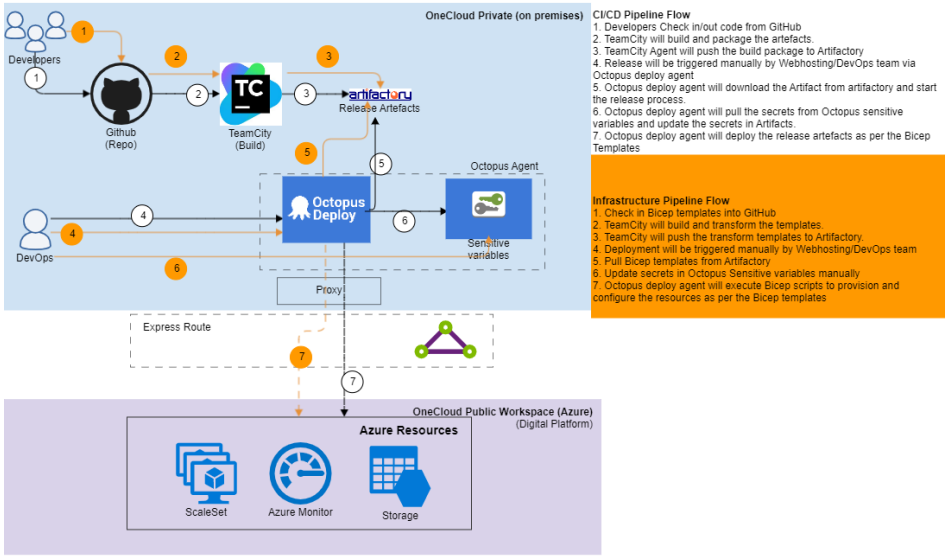


As part of the migration of FirstNet web from ODP IF to One Cloud, we also need to finalise the strategy for migration of configuration files from current ACDC tool to the new deployment tool - Octopus deploy. The strategy should be in line with our future state goal. However, we need to consider the time constraints with respect to completion of migration activity.

Current deployment flow

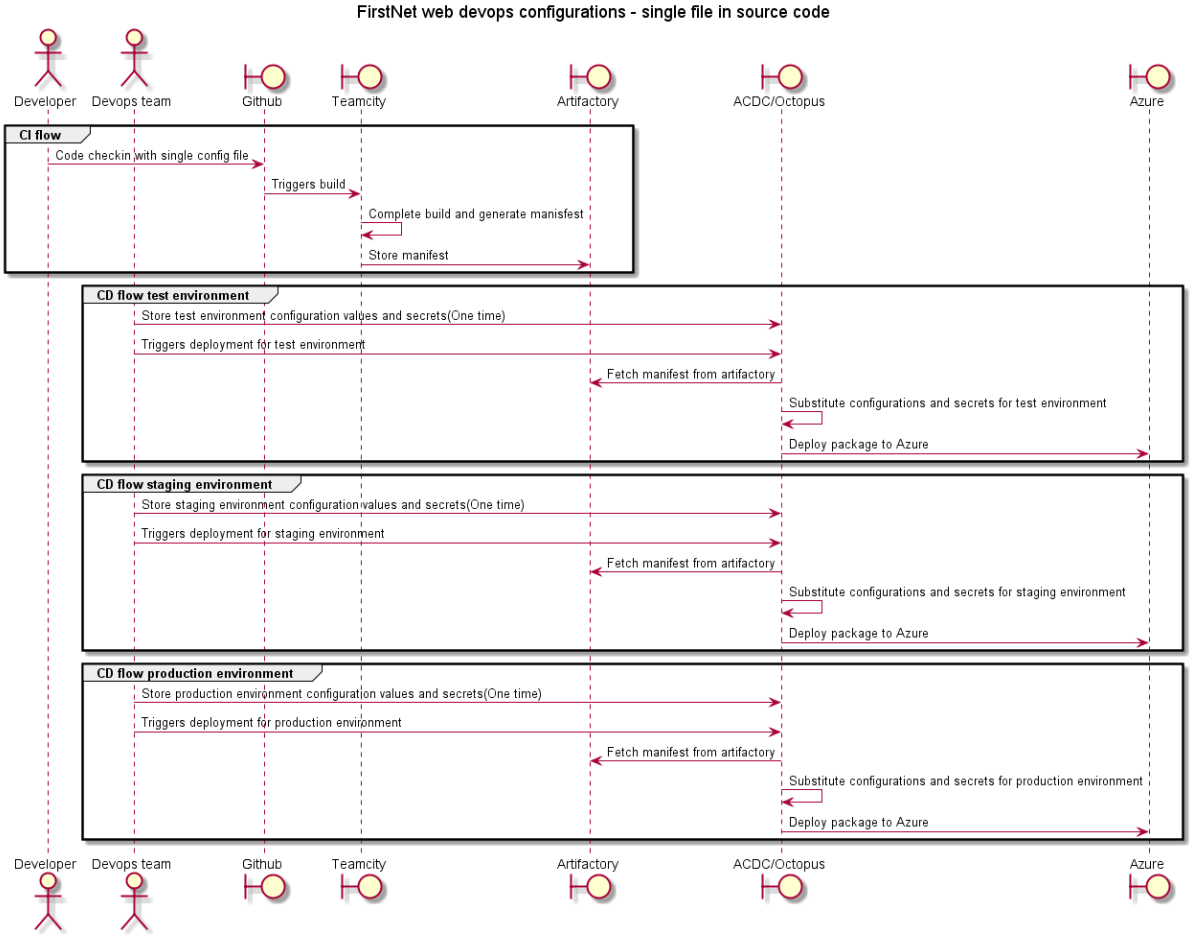


Proposed deployment flow after migration



We have selected below options as possible approach to migration of configuration files.

Option1 - Configuration in single file for all environments as part of source code (Recommended)



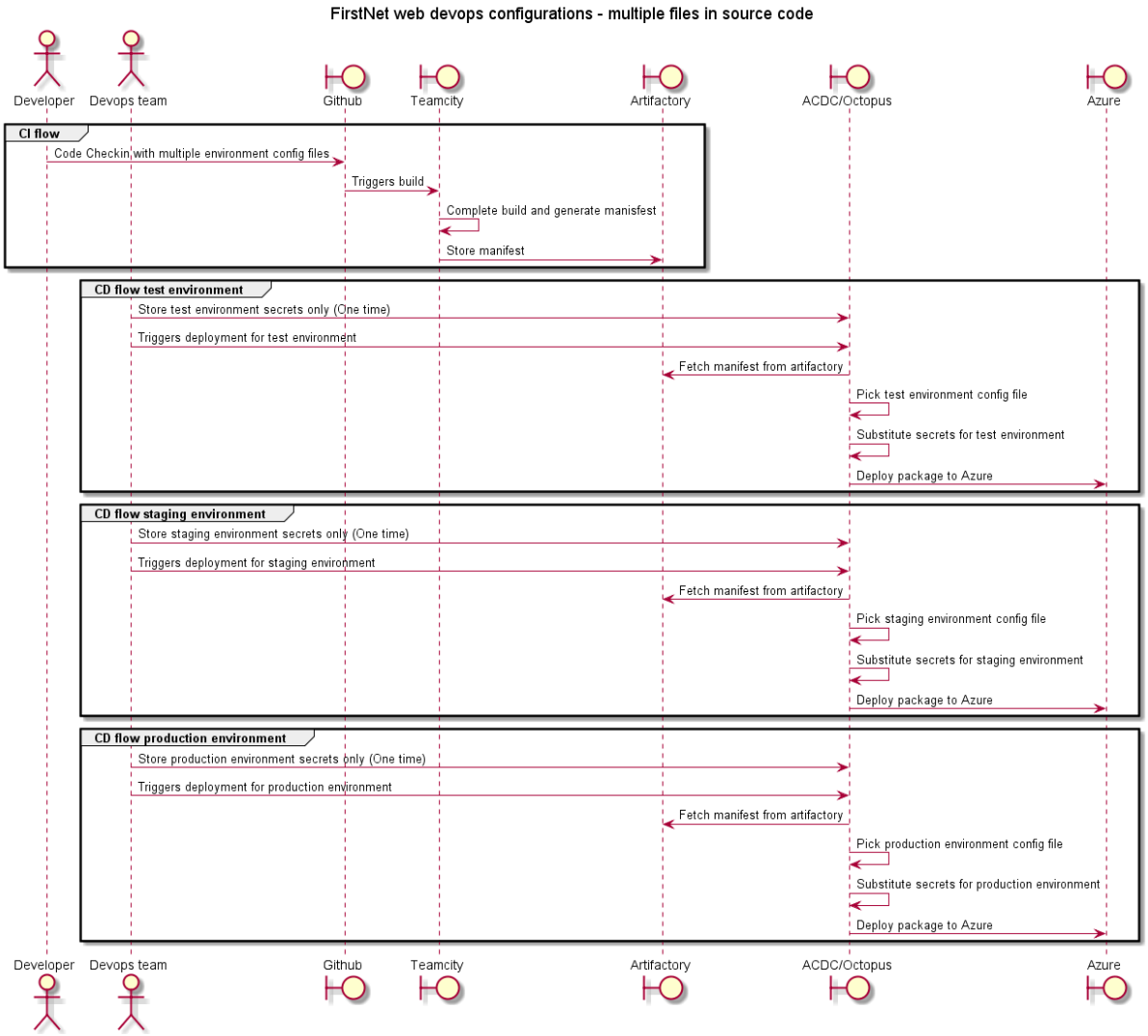
Pros

- Consolidated environment specific configurations values
- Less coupling with DevOps tool
- Can manage/add/remove all environments from single place
- Less developer dependency
- No code changes required for configuration changes
- Relatively less changes in existing code and release process to implement this as part of migration, resulting in shorter release cycle

Cons

- Configurations are segregated from source code
- Need to involve DevOps team to add/remove config values

Option 2 - Configurations in different file for each environment as part of source code



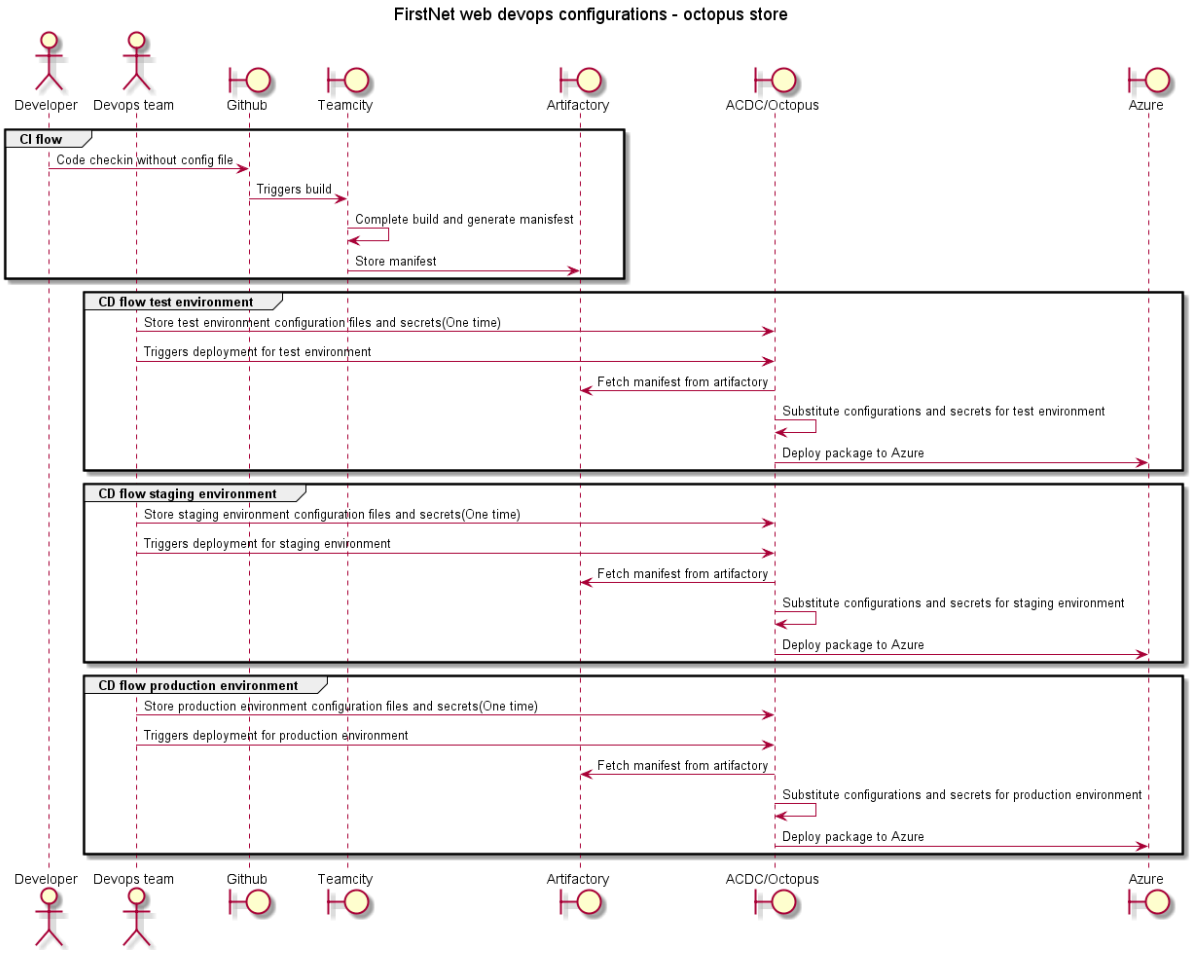
Pros

- Configurations are co-located with source code
- Configurations changes can be done by developer independently

Cons

- Segregated environment specific configurations can be difficult to manage from within source code
- More developer dependency which can slow down environmental changes or release process
- Need code update for configuration changes
- Need understanding of all environments by Development team
- Relatively more changes in existing code and release process to implement this as part of migration, resulting in long release cycle

Option 3 - Configurations in single file for each environment as part of DevOps tool(ACDC/Octopus)



Pros

- Consolidated environment specific configurations values
- Can manage/add/remove all environments from single place
- Less developer dependency
- No code changes required for configuration changes
- No changes in existing code and release process to implement this as part of migration

Cons

- Configurations are segregated from source code
- More coupling with DevOps tool
- Need to involve DevOps team to add/remove config values

Status

Due to the use of dual tools (ACDC and Octopus deploy) during the interim state of FirstNet web migration(refer SoaP) the impact of configuration migration has increased significantly and the change will require extra regression effort. Considering the time constraints as we have a fixed deadline, the configuration migration will be carried out with option 3 for now.

Deployment templates

Octopus out of the box doesn't support declarative templates to configure pipelines. There is an option to use Rest API to configure Pipelines using supported programming languages. However, considering the migration timeline and future state of Devops pipelines (Post separation from CBA), it has been decided to configure the Pipelines directly in Octopus. However, the ARM templates and any custom script file used to configure Infrastructure or Applications will be part of separate DevOps repository.

Team city builds will be used to pull templates from GitHub and upload them into Artifactory. These templates will then be consumed by Octopus pipeline.

Programmatic deployment templates

Octopus supports programmatic way of defining the pipelines using APIs. You can create your pipelines using different programming languages and calling Octopus apis.

Pros

- Pipeline code can be reused across different Octopus environments.
- Pipeline code can be completely source controlled.

Cons

- Considerable upfront effort
- Since Octopus is a stop gap tool till CFS separates from CBA, this effort will mostly be technical debt

Manual pipelines with Arm templates/Biceps 

In this approach, all the templates and script files will be stored in GitHub repo. The pipeline configuration will be done manually in Octopus directly. We may use nested projects to configure some of the common functionality. In nested projects , one Octopus project can call other projects, and can pass in parameters that can be referenced in the child project. This can enable re-use of a sequence of deployment steps (e.g. deploy virtual app, deploy configuration file, run a script) that only vary by value.

Pros

- Pipeline code can be partially reused across different Octopus environments.
- Pipeline code can be partially source controlled.
- Relatively faster to implement.

Cons

- Lack of complete reusability.

Using Terraform plugin

Octopus also supports configuring pipelines using terraform plugin. You can define your pipelines in terraform and execute them via Octopus plugin.

Pros

- Pipeline code can be reused across different Octopus environments.
- Pipeline code can be completely source controlled.

Cons

- Considerable upfront effort
- Since Octopus is a stop gap tool till CFS separates from CBA, this effort will mostly be technical debt

Auto scaling

Autoscaling will be handled using deployment bootstrap scripts as part of the deployment package.

Auto Scaling via tentacle client

In this approach, the application will be deployed using Octopus tentacle functionality, where in the application package first deployment and any subsequent auto scaling will be handled via Octopus deploy.

Pros

- Relatively easier to setup

Cons

- Dependency on Octopus for autoscaling
- Require continuous connectivity between Azure and Octopus tool

Autoscaling via bootstrap script 

The application will be deployed as a deployment package to blob storage and have the VMs pull and execute this package on startup using a script extension. This approach makes sure that any subsequent autoscaling is handled at Azure workspace level and there is no involvement from Octopus post deployment.

Pros

- No dependency on Octopus for autoscaling
- No continuous connectivity between Azure and Octopus tool
- Relatively faster autoscaling

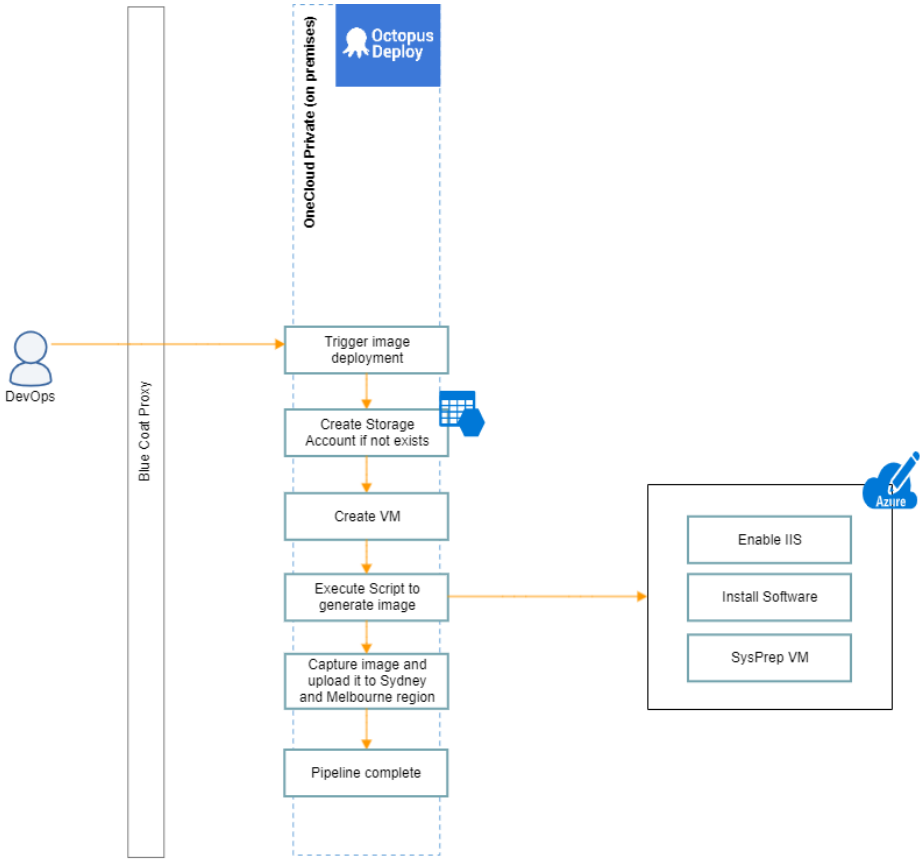
Cons

- Relatively higher upfront effort

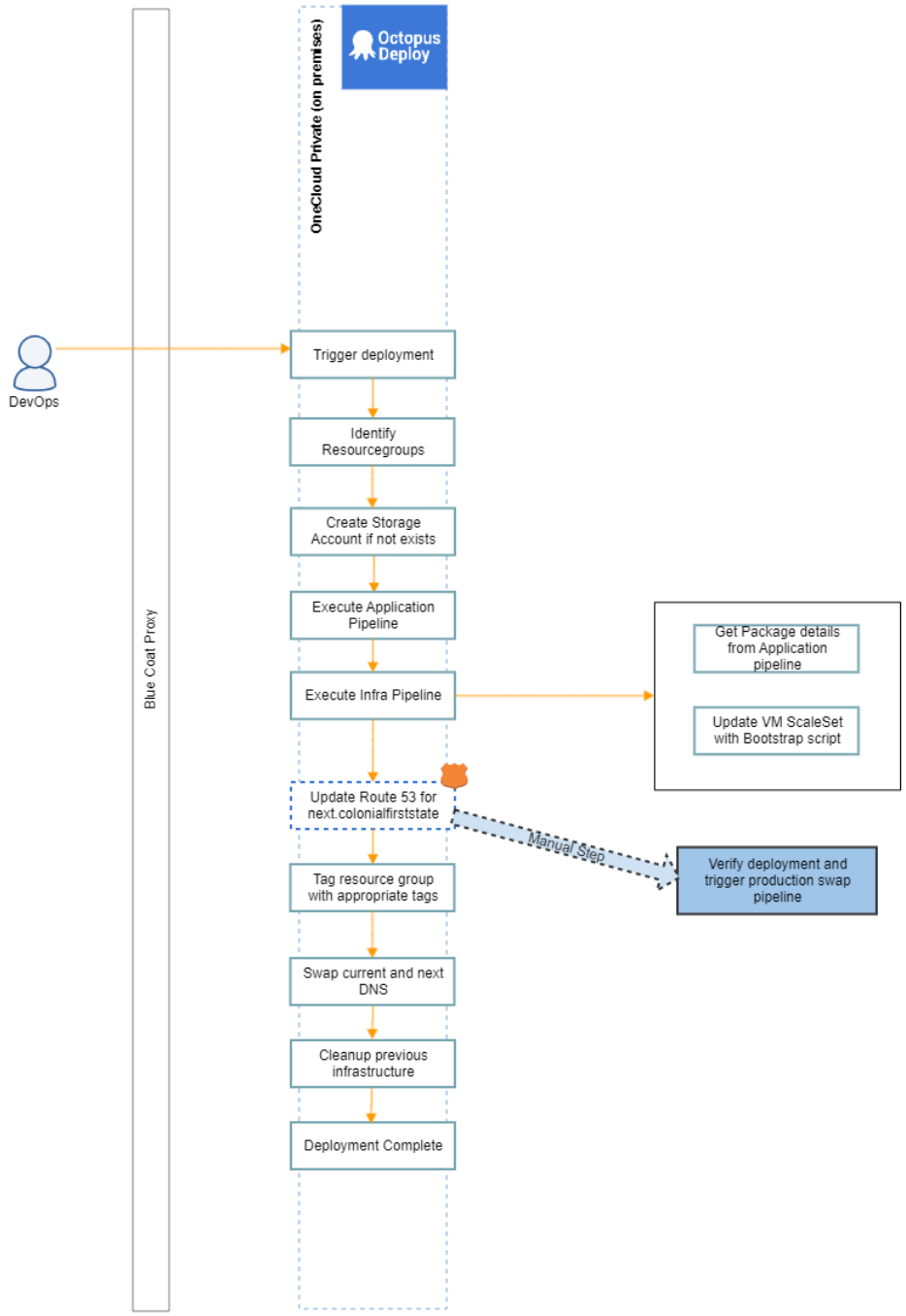
Firewall rules

To be added.

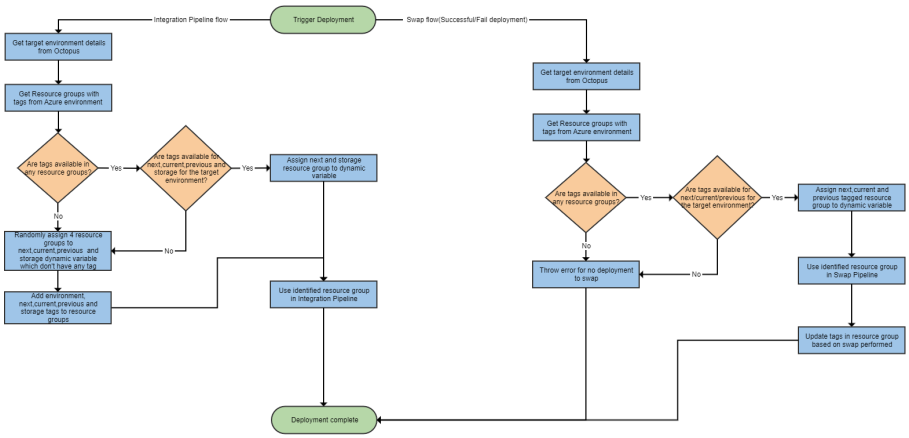
Base Image Pipeline



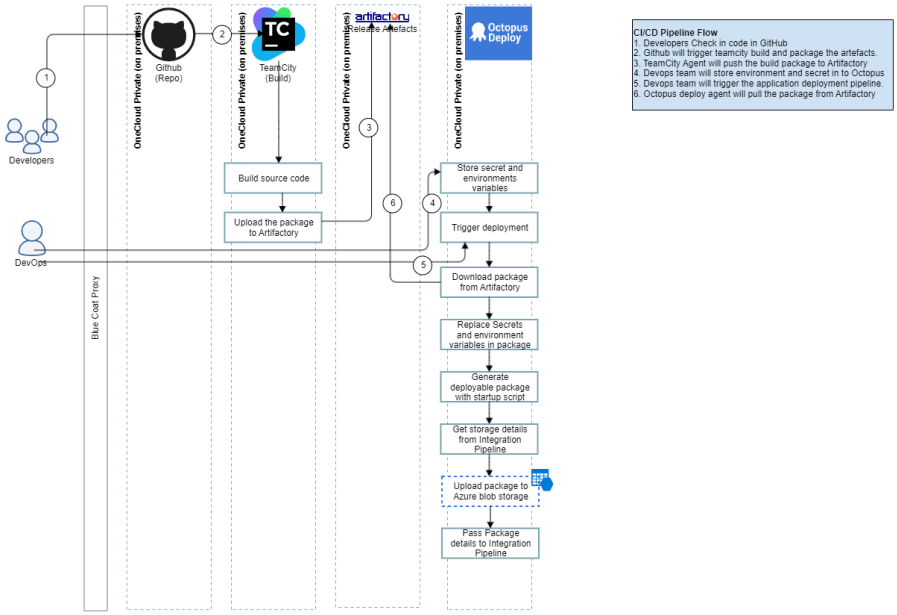
Integration Pipeline



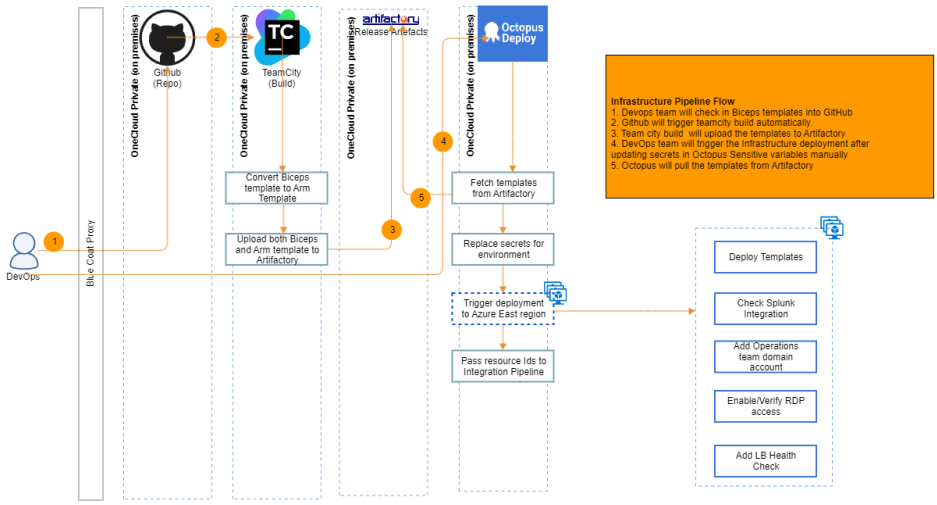
Resource group Tagging Flow



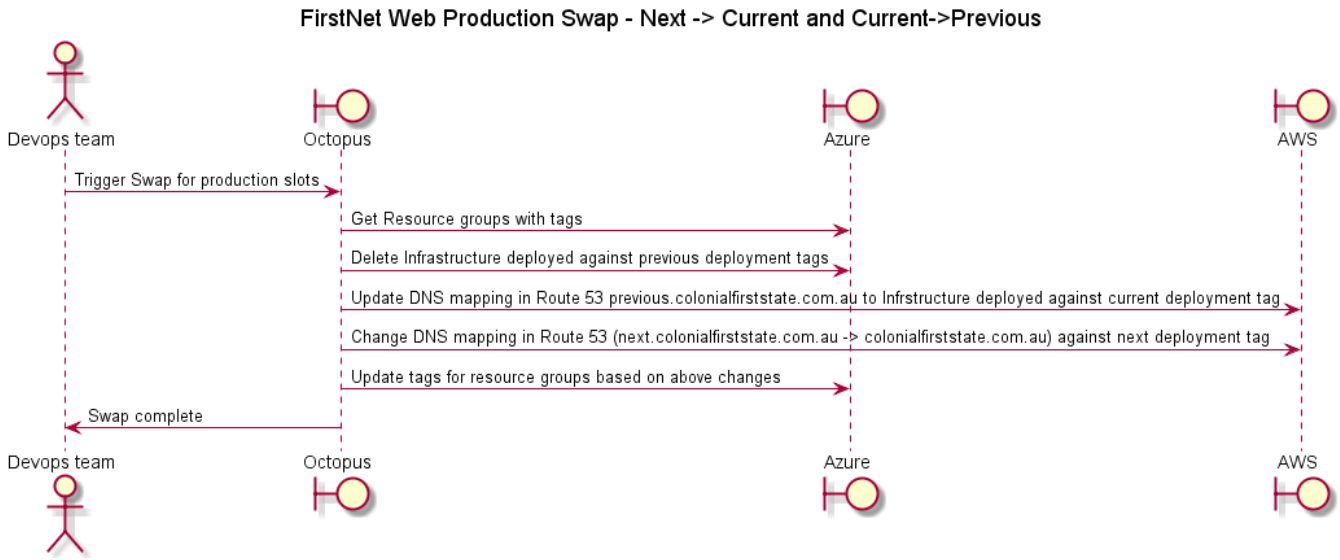
Detailed Application Pipeline

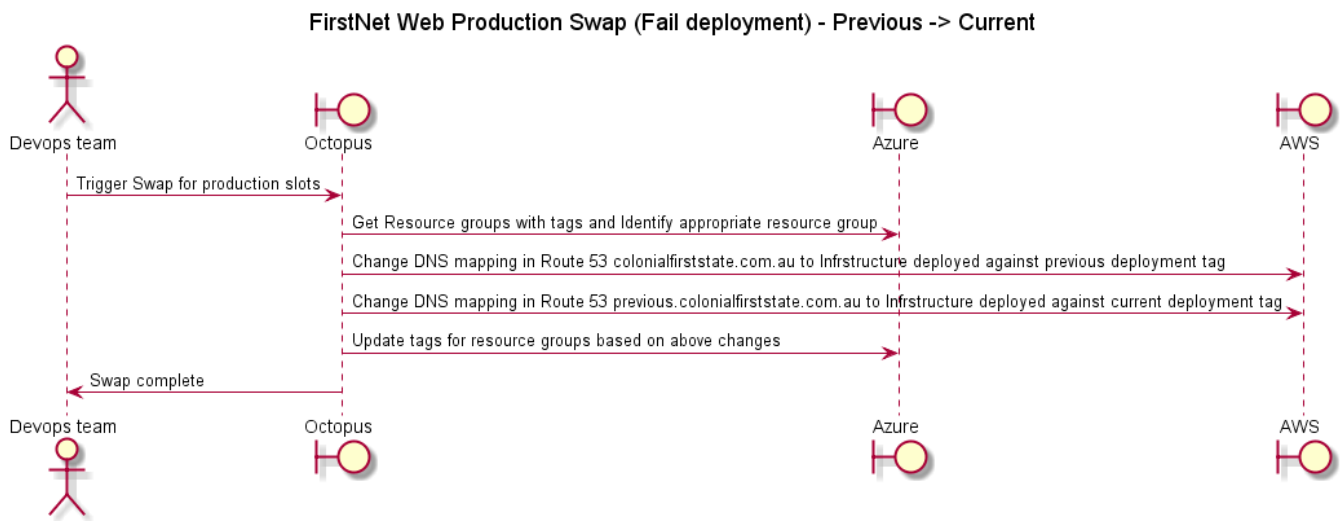


Detailed Infrastructure Pipeline



Production swap pipeline





More details here : FirstNet Web - Blue/Green Deployment

DR Pipeline