**Azure DevOps**

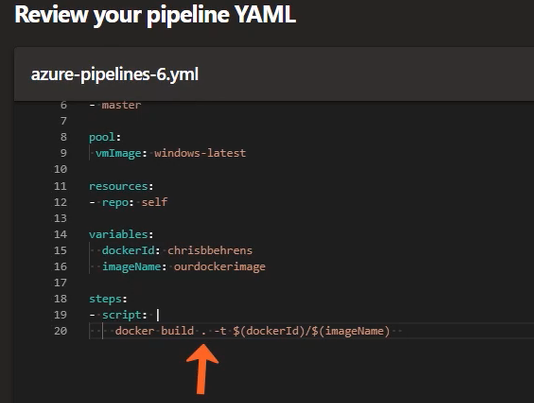
**Boards** – Where you manage human beings doing the work

**Repos** – where code lives

**Pipelines** – where you build your code

**Artifacts** – where you package 7 store code

**Containers** – The h/w is abstracted into a docker file. Create an image with a build. Push to a container registry, Use DocketHub or Azure Container Registry.



**Dockerfile** –

The base docker file content is simple FROM Microsoft/iis

Review & edit YAML

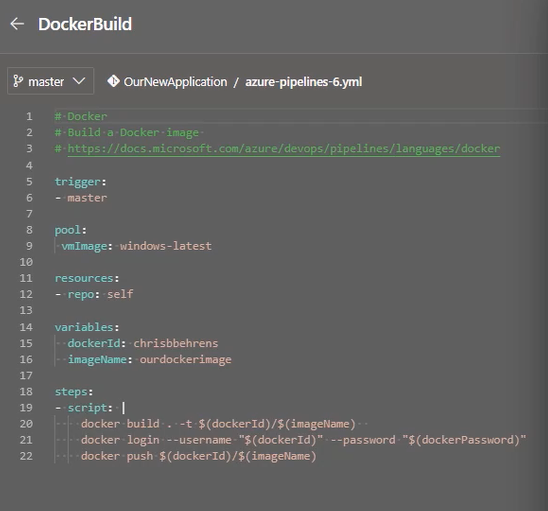
The password needs to be specified

**Dockerfile** –

The base docker file content is simple FROM Microsoft/iis

Review & edit YAML

The password needs to be specified



**Setting up notifications for when builds fail** Is done in the devops portal under Project settings -> Notifications -> Delivery settings . Notifications can also be sent to Microsoft Teams.

Click the + & Add what notifications you want to subscribe to. You can set up nottifications for conflicts etc.

**Package Deployment formats**

Zip, - Simplest

Nuget – Octopus Deploy

XebiaLabs – DAR Files – requires a manifest. (Needs XL deploy extension from Azure Marketplace)

**Octopus Deploy** – Uses Nuget . you need the Octopus Integration extensions for Azure which you can get from Azure Market place. (called **Octopus Deploy Integration)** Specify Octopus Deploy server. & the API key

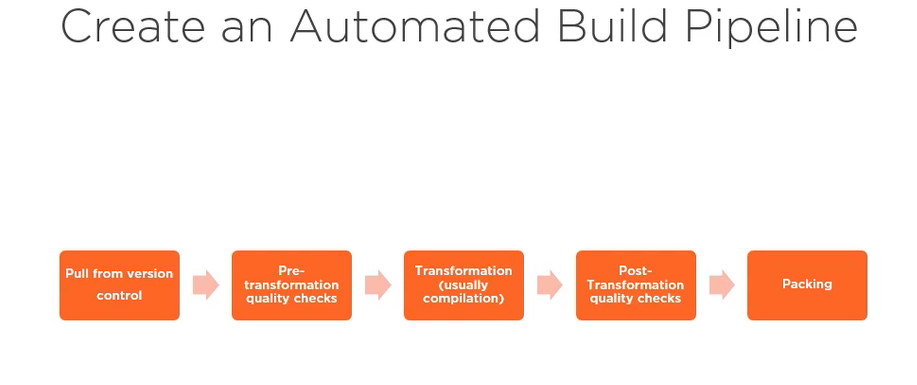
**Build** **Pipelines**

Build should fail if (compilation or tests fails)

Octopus Deploy – Uses Nuget

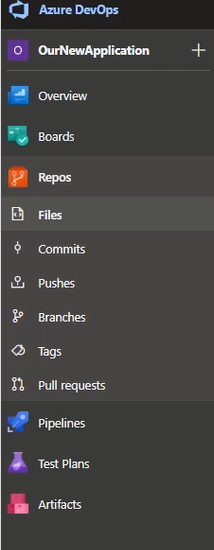
XL DAR file ???

Divide into automate today & Automate later steps.



Sign in to Azure devops at dev.azure.com. Create a new project & set the privacy levels to those you want. Select your version control (Teams or Git etc). Select Agile. The Beaker icon – Test plans

Artifacts –



Create new repo

Use git-commit & git-push to add your code to your repo

In pipelines you will see Connect , select, Configure & review.

The pipleline review is YAML

Click save & run.

It will prepare an agent for the job.

The logs will show if it has succeeded

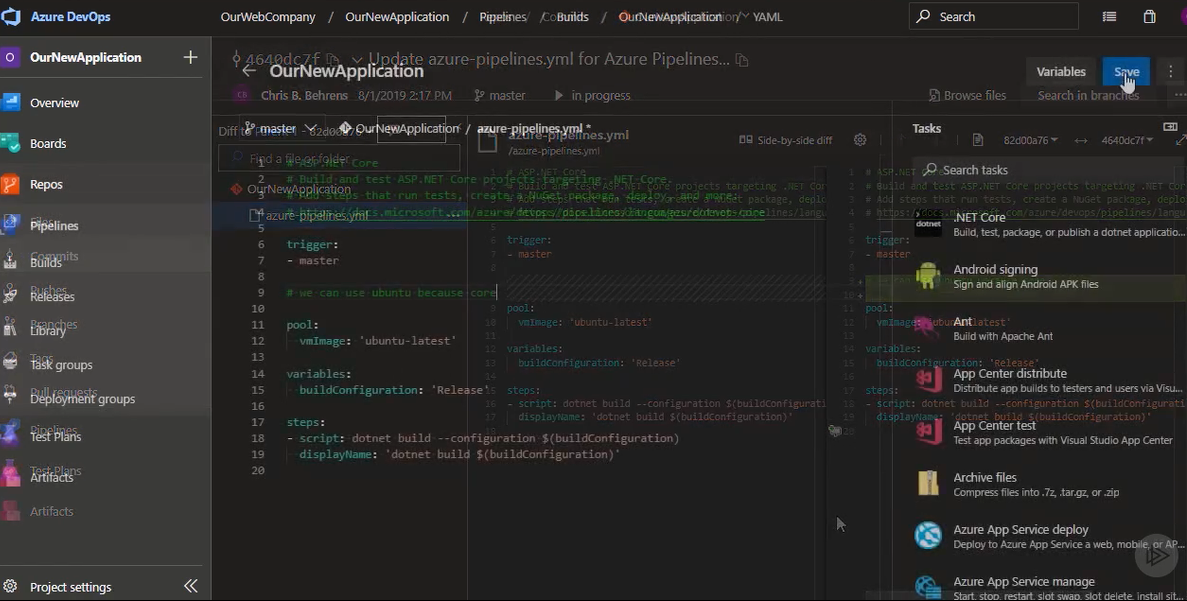
**YAML – YAML aint a Markup Language**

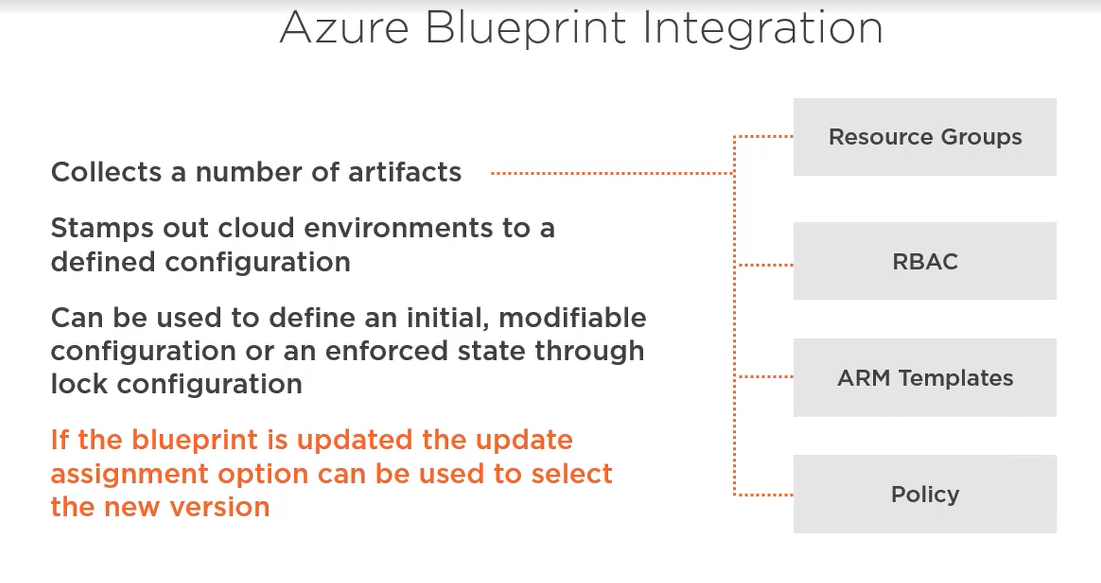
Build allow revertability

Dev (feature branch) -> Staging (develop) -> Production (master or release branch)

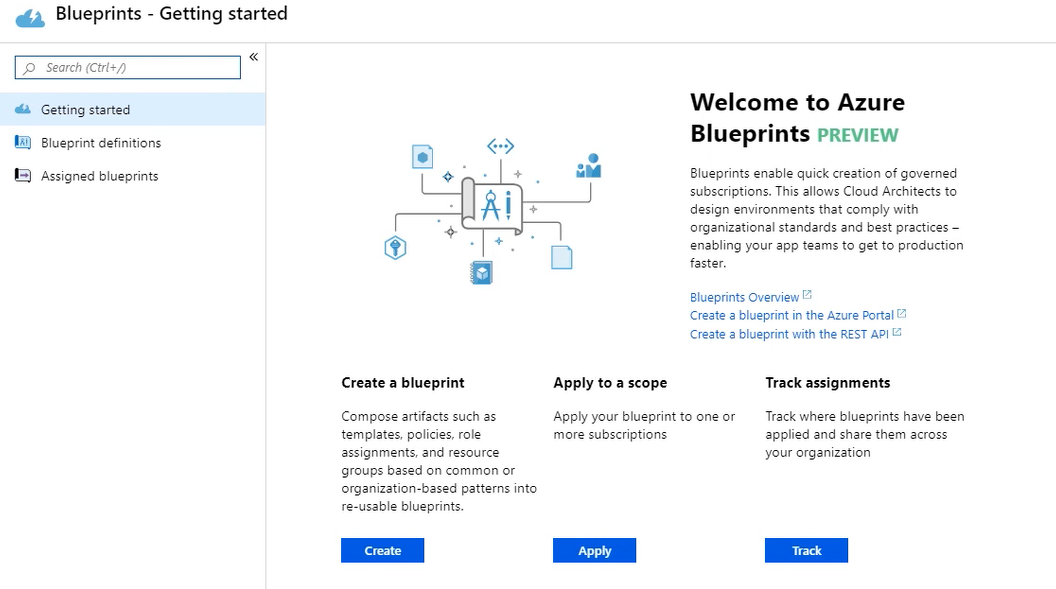
**Steps for New project;**

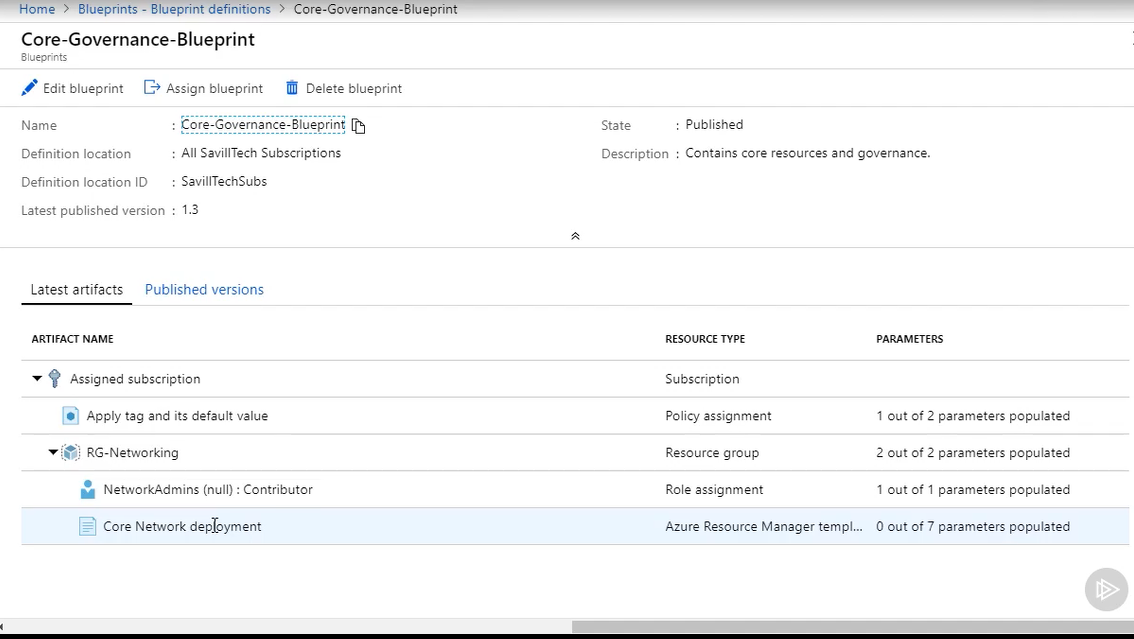
1. Create new library
2. Commit to local branch
3. Create modified build definition which includes new project
4. Commit to develop
5. (if hotfix needed from testing, create a new version of build def that does not include new project & works correctly)



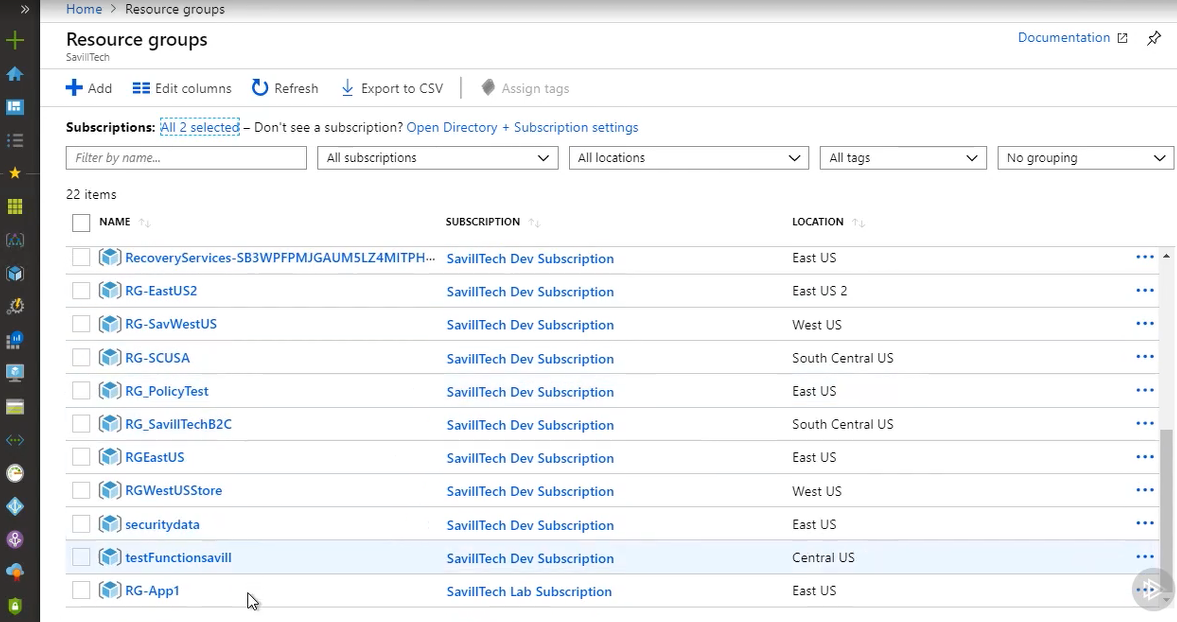


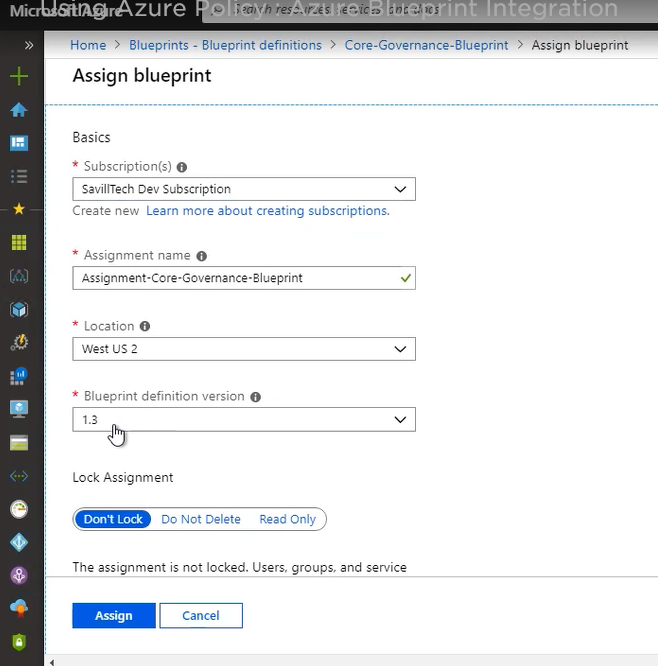
Blueprints can be saved either to management group or subscription. In the portal select blueprint

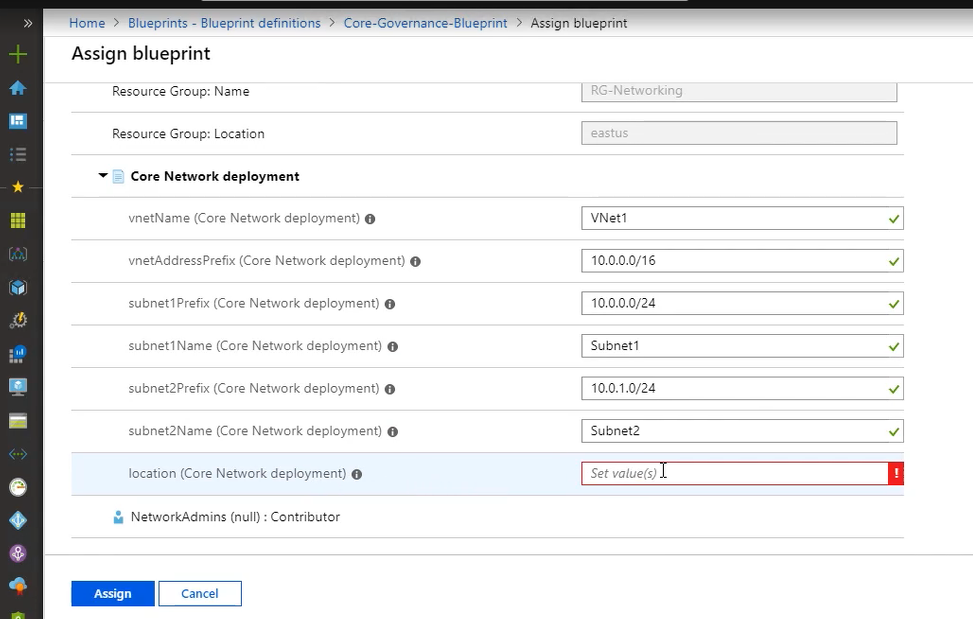


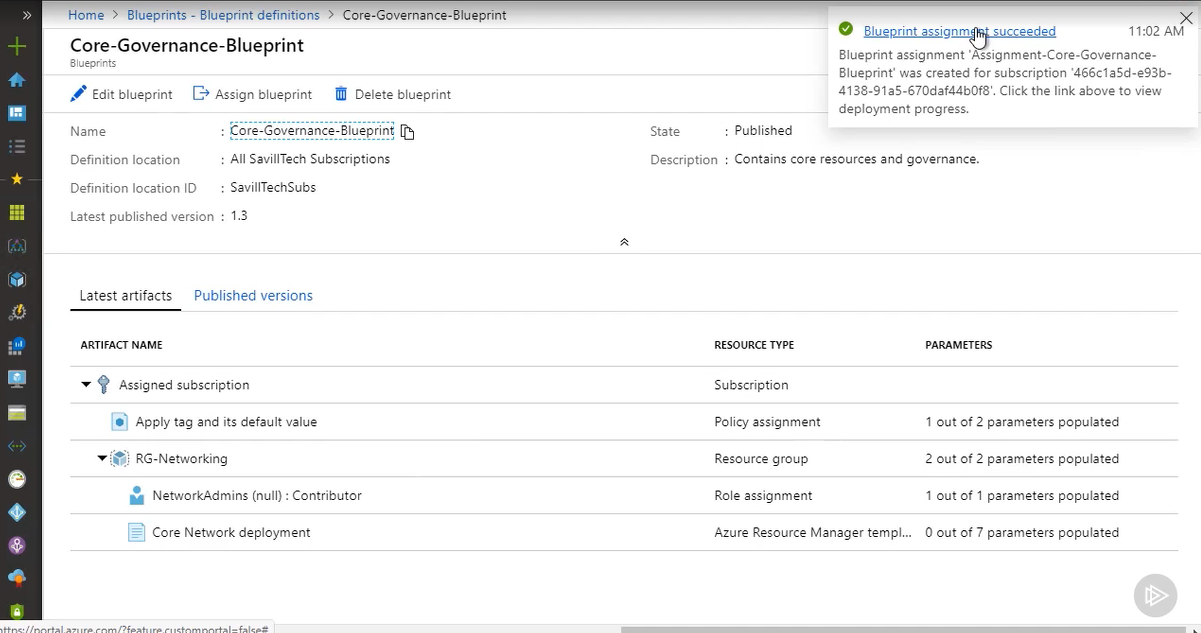


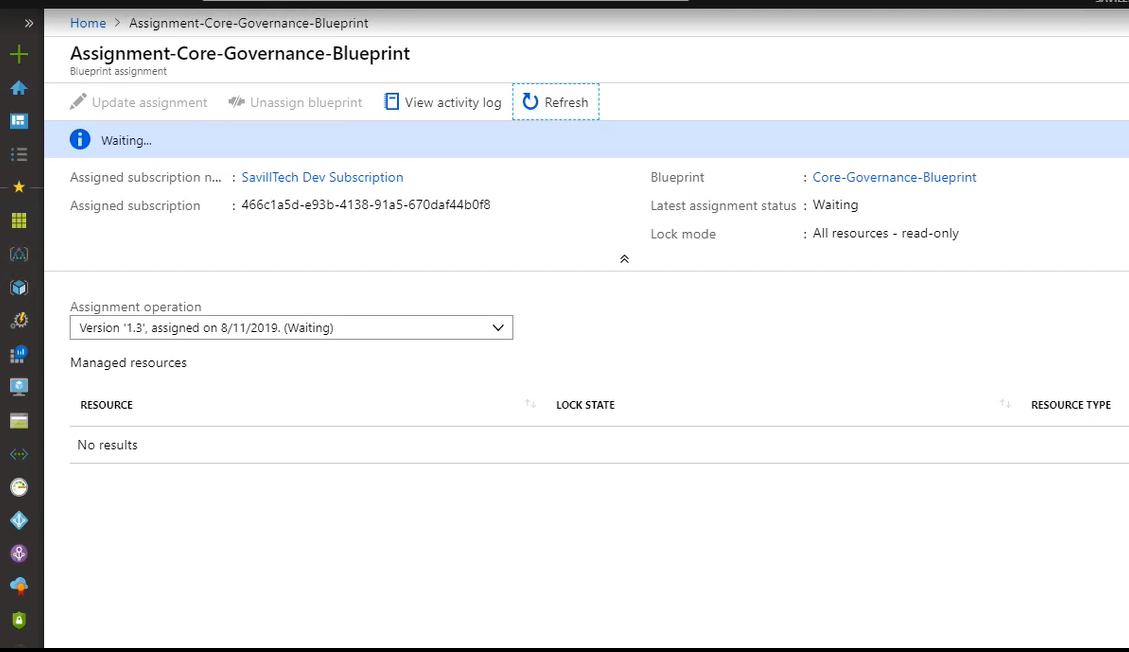
Select either existing or new Resource group









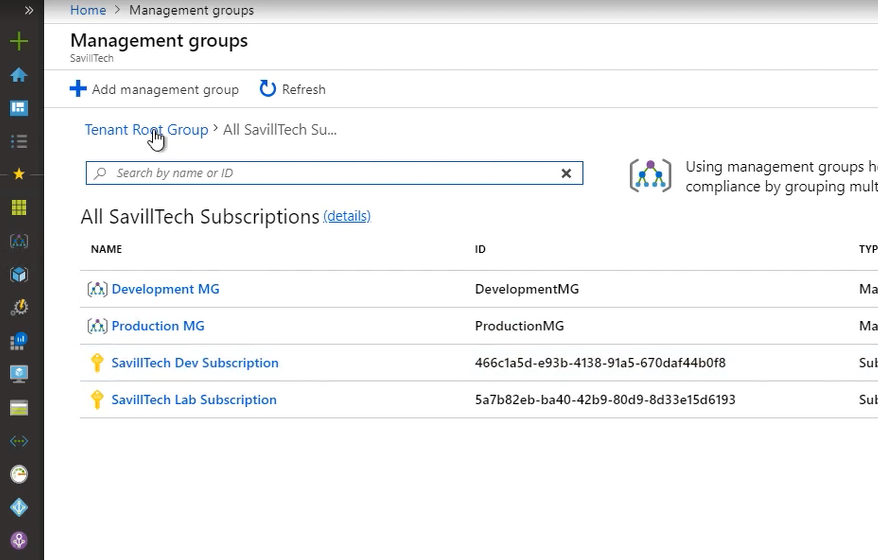


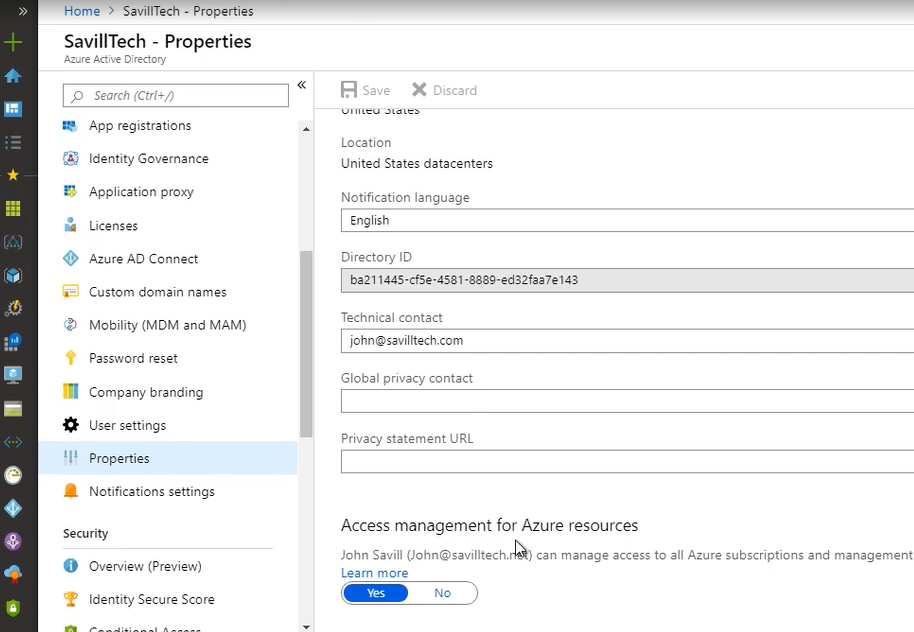
Note:- if you delete the blueprint, it will not delete the resources created by that blueprint.

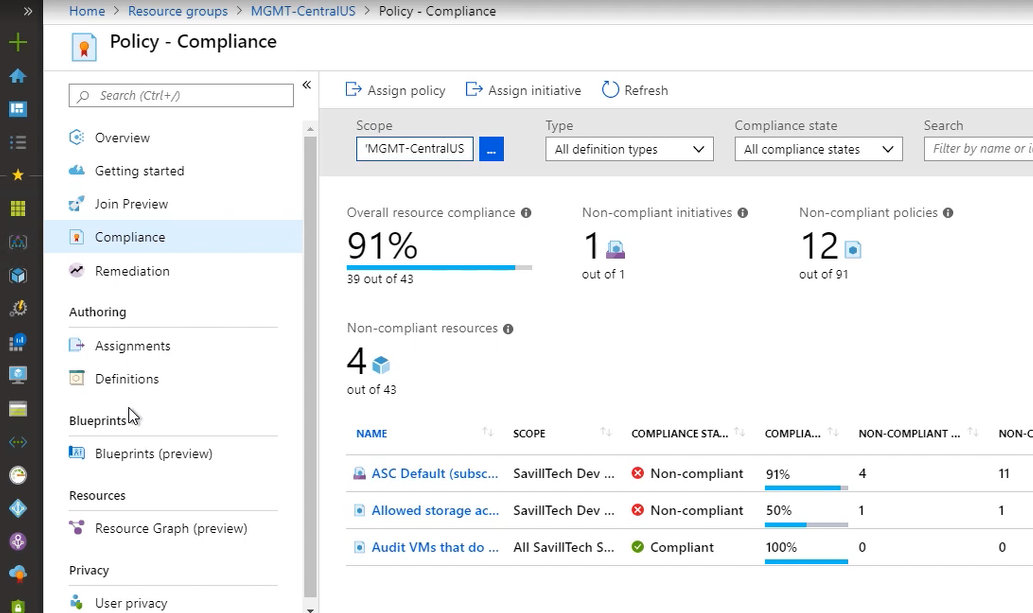
**Pipelines & Policy. Policy Fundamentals:-**

Scope at management group, subscription or resource group levels.

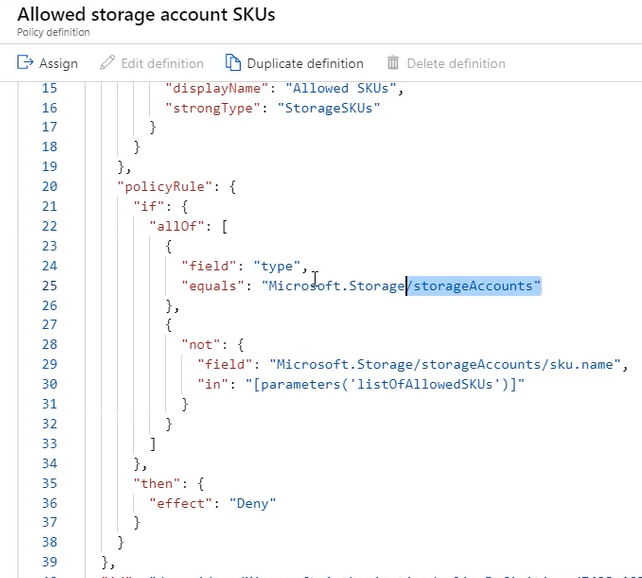
As you get closer to the resource group you may want to scope at more specific levels e.g. pre prod & prod levels etc. (Root Management groups i.e. Tenant groups could have a broader policy)

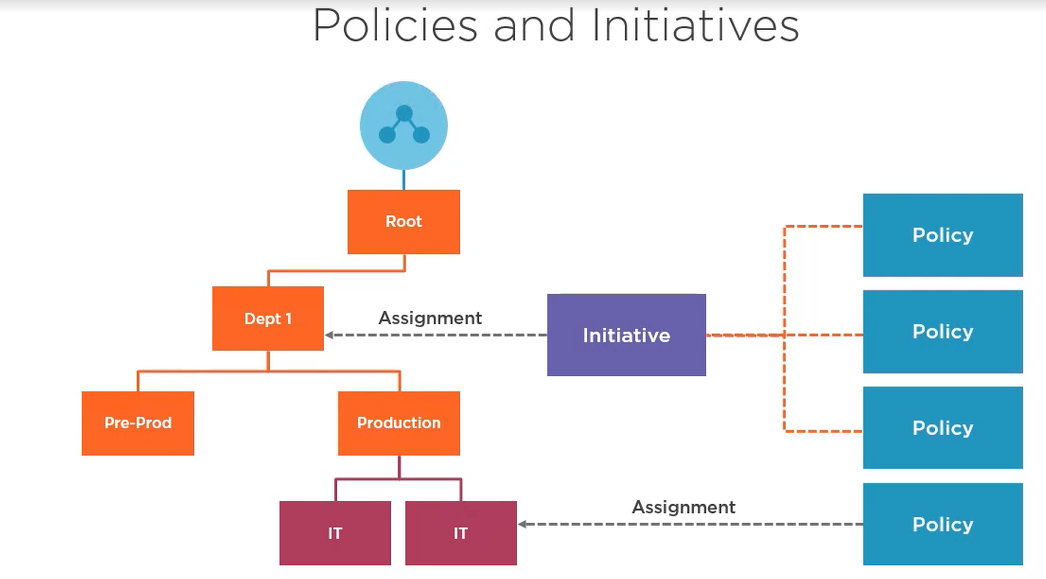




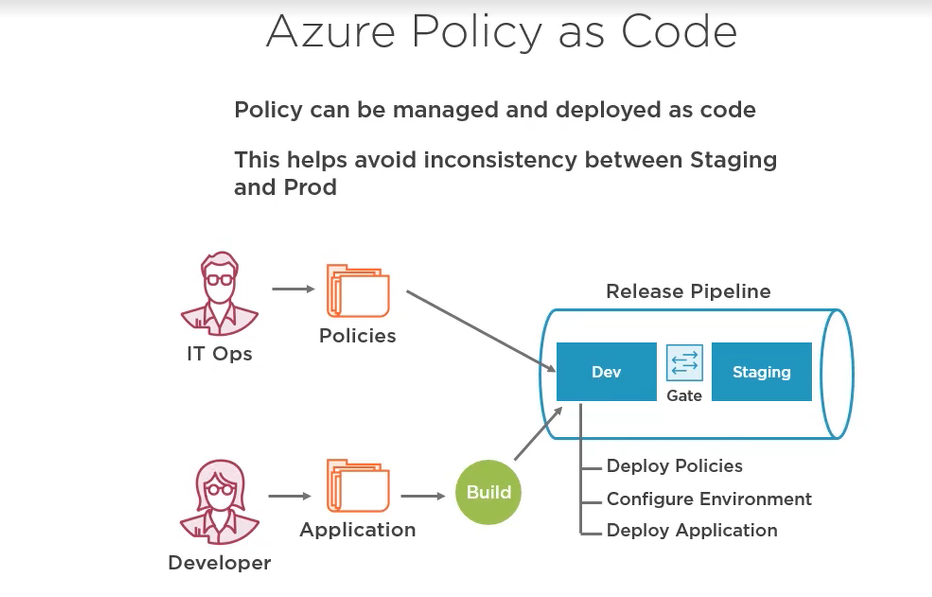


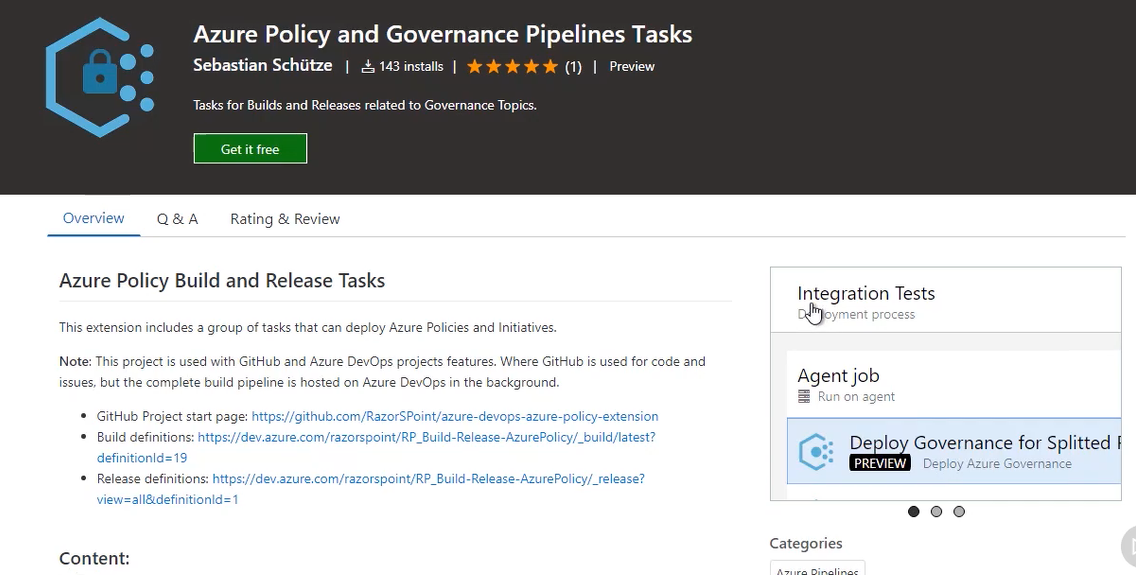
You get more restrictive the closer you get to the actual resources.





In the above example there would be no benefit because the policy is already in the initiative that is applied at a higher level. The policy has parameters & could have diff parameters depending on the level. SO you generally want to make policies more general at higher levels & more restrictive the lower down you get when you get closer to the actual resources.





Either use this or use the policy from github called batch-policy-scripts

