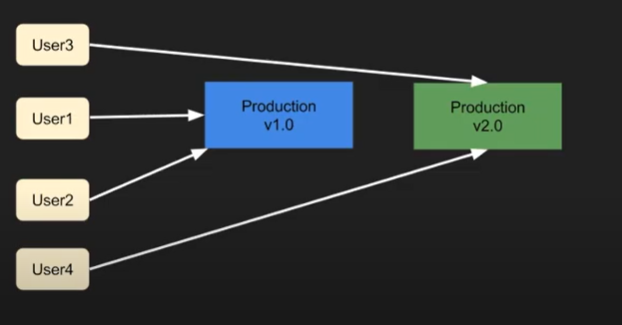
**Blue Green Deployment**

* Design pattern or technique with which we can reduce the downtime of an application
* And by running two application which are identical in the production environment
* 
* Blue is the current version of the application and green is the future version
* Zero down time
* We start redirecting traffic from blue to green slowly
* After we move all the customers, we are going to remove the blue instance
* In cloud foundry we have a middle layer of router which will redirect users to the blue or green deployed version

**Cloud Foundry Blue Green Deployment**

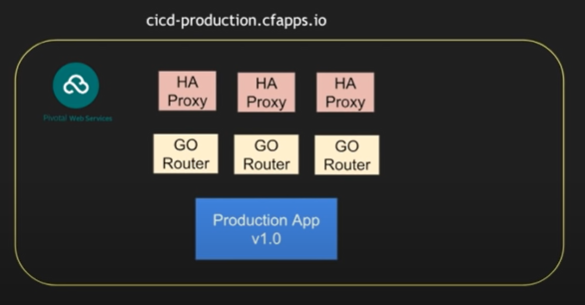
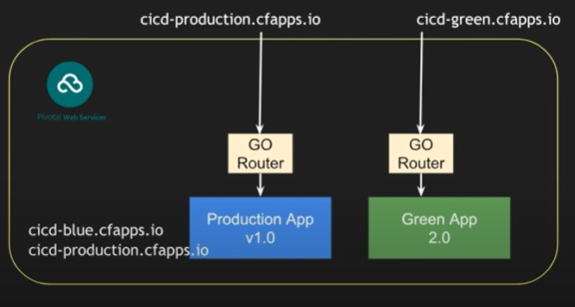
* Cfapp.io – where the application is hosted
* Cicd-production – host name of the application
* Cloud foundry maintains three different IP addresses which points to the same application
  + If you do a ns lookup on cloud foundry you will find three IP addresses
  + Load balancers
  + Go router has the mapping from GO Router to Production App
  + 
  + 

Figure 1:Step 1

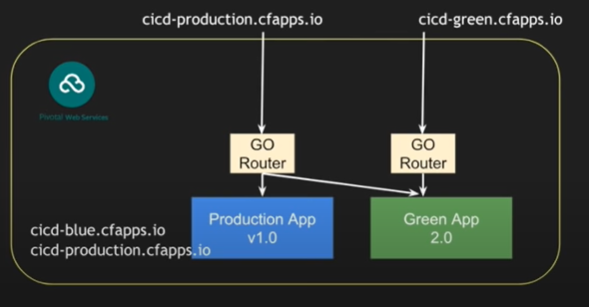
* + 

Figure 2: Step 2

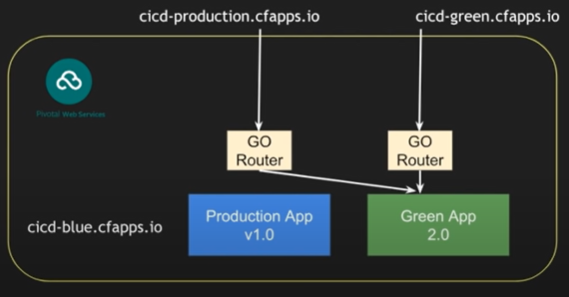
* + By default pivotal cloud foundry runs a round robin fashioned routing
    - Some requests will be served by blue instance others by green instance
  + 

Figure 3: Step 3

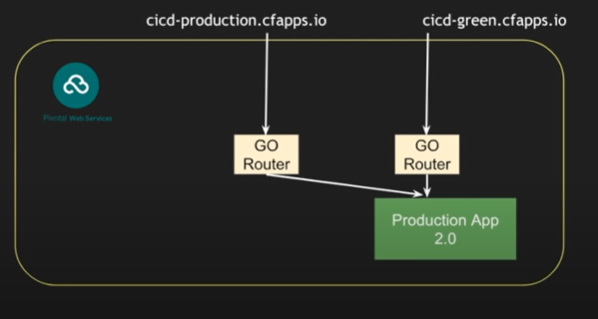
* + 

Figure 4: Step 5

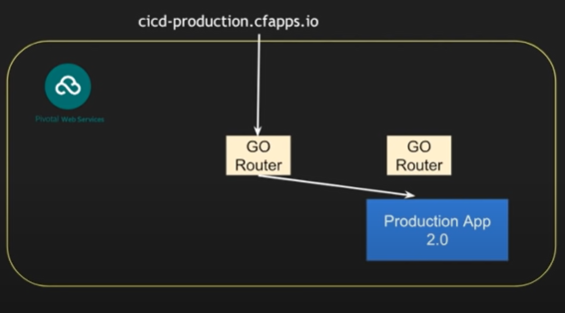
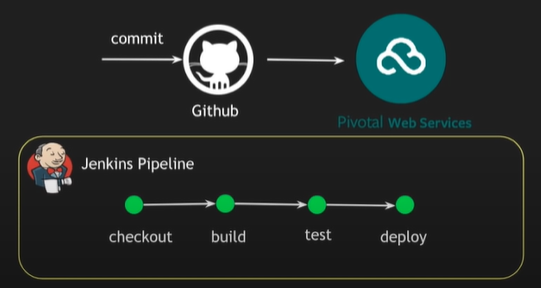
* + 

Figure 5: Step 6

* + Zero downtime: Go Route is going to map every new request to the green app
    - If some requests are stuck with the old route it will not kill the request until it is been served

**CICD using Jenkins Pipeline**

* 

**Cloud MTA Build Tool (MBT) (https://sap.github.io/cloud-mta-build-tool/)**

* Stand alone command line tool that builds a deployment ready multi target application (MTA) archive (.mtar) from the artifacts of an MTA project
* According to projects MTA development descriptor (mta.yaml)
* The build process and the resulting multitarget application archive depend on the target platform
  + Currently supported platforms:
    - SCP:
      * Both Neo and Cloud Foundry
    - SAP HANA XS advanced model
* The results of each module's build are packaged into an archive together with the multitarget application deployment descriptor
* All resource definitions are passed to the mtad.yaml file as is without mapping and validations
* How to build an MTA archive from the project sources
  + You can use one of the following two approaches:
    - One-step build using the mbt build command
      * Generates a temporary Makefile according to the MTA descriptor and runs the make command to package the MTA project into the MTA archive
    - Two-step build using a combination of the mbt init and make commands
  + Both methods leverage the GNU Make technology for the actual build