**:New app onboarding procedure**

**Initial settings.**

1. **Go to secrets manager**, then UC2s-ODIN-MISSION\_PROD, then retrieve values, then scroll down and click on “add-row”. We typically use the same format (<appname\_RDS\_DB> for database ID and <appnamedb> for password). then click on “add row” again, use this format (<appname\_RDS\_ROOT\_USER-appnameadmin> and Odinad….Pass.. for the root user password). open a note pad and copy all the above key:value pairs for future use when you create the actual RDS. When you create the data base you will also need to copy its endpoint URL to your notepad for future use when you will modify the db\_admin\_creds file.
2. **Create the RDS database**. We mostly use MySQL or PostgreSQL). Go to RDS, then click on “create database”, then choose the engine type and version. Modify the DB instance identifier to me <odin-mission-prod-appname>. The “Master username” will be < appnameadmin> and the “master password” will be < Odinad….Pass..>. select the “burstable classes” option, then choose the size (t3.small for example), then choose the “storage type (gp2 for example), then turn off autoscaling by selecting “do not create a standby instance”. Change the VPC to be the mission prod one (odin-mission-prod tenant VPC), for security groups choose from the drop down menu the corresponding one. For example for SEWOl, since we used a postgresql engine, the security group was MISSION-PROD-PSQL. Get rid of the default security group. Click on “additional config” and make sure the TCP/IP port is 5432. Scroll down to the other “additional config” for the “database options”. for database name, put <appnamedb>. Note that if you ever want to delete the database, you must go into the parameters/settings, scroll down and disable the “deletion protection”.
3. **Keycloak configuration**. First create a “group” for the app. To do that go to “groups”, then “platform one”, then “mission apps”, then click on “create group”, and name it <appname>. Search for the newly created group and click on it. Then go to your browser and copy the last few characters of the URL (form the last character to the character next to /). Put it into a note pad. Now go to “clients”. Create a client with the client ID in this format <mission-appname\_the URL characters\_prod> and the name <appname>. This is how keycloak will identify which group a member belongs to. Hit next and select “client authentication”, hit next, then on “valid redirect”, input [https://virtualserviceofyourapp.odin.dso.mil\*](https://virtualserviceofyourapp.odin.dso.mil*) and [https://virtualserviceforyourapp.odin.dso.mil/api\*](https://virtualserviceforyourapp.odin.dso.mil/api*). Then save it. Now go to “credentials” then view and copy the “client secret”, put it into notepad. Also grab the URl and paste it into your notepad. Add yourself as a member of the group (choose the group, then “members”, then “add member”, then search for your name and select)
4. **Go to the Odin-mission-prod build box to build the auth secrets**. Assume “git” role (sudo su – git). Navigate to sipr-mission-bootstrap/bigbang/core/envs/prod/secrets/auth\_configs/ , then duplicate a file and change its name to the name of your app, for example do this ( cp jfac-auth.enc.yaml sewol-auth.enc.yaml to duplicate the jfac file and changing its name to sewol), then vim into the newly created file to perform the necessary modifications. change the “prefix” attribute to < virtualserviceofyourapp.odin.dso.mil> (use the vim command :%s/ oldpattern/newpattern/ g , the /g is for global. Use it to make the change everywhere) to make it easier. Also change the “client\_id” and replace all the old app name with the name of the app you are working on. Also change the “client\_secret’ to the client secret you grabbed from keycloak.

**Cd** back up two directories to prod/ in there vim patch.bigbang. then perform some modifications: try to keep it in alphabetical order. So search for an app that starts with the same letter as your and add your app key-value pairs next to it. Again, use the vim commands to make it faster (2yy to copy the current and next line, p to paste them, cw to change a word, or cl to change a letter, etc…) add this.

**- kind: secret**

**Name: appname-auth**

Then run a “kustomize build . “to make sure it build and confirm the formatting done above was correct. If there is a mistake, then the kustomize build command will fail to run.

**Mission bootstrapping**

1. Get access to the manifests repository of the application depending on the impact level (il) they are on. For example, tiger ranch is on il4.
2. Navigate to our sipr-mission-bootstrap repository and do a “git pull” to catch all the changes made on the master branch.
3. Create a new branch usgin “git chechout -b <name-of-the branch>”.
4. The last step is to bundle your branch and send it over an AWS S3 bucket. Do this (git bundle create sipr-mission-bootstrap-appname-yyyymmdd.bundle <name-of-your-branch>)

**App manifests and integration manifests updating**

1. After the bundle is created and pulled into the odin mission prod build box by assuming the maintuser role. then you need to unbundle it by doing this (git clone -b <name-of-your-branch> <the bundled-file>). Then from the directory where you unbundled it, there should be a folder with a similar name without the .bundle extension. That is the file you will want to work with next.
2. Do “mv <name-of the-file> <appname>” to rename the folder to appname. Then do “mv <appname> <DO-NOT-DELETE/>” to move it into the DO-not-delete directory.
3. Assume the git user role, then cd to sipr-mission-bootstrap/integrations/plugins/ . then do a pwd to capture the path. Then exit the git user and from maintuser, cd into appname/integrations/plugins. From there do “sudo cp -R s3 <path-captured-from-pwd>” to copy recursively the S3 directory to the git path. Then do “ chown -R git:git path-capture-from-pwd/s3” to make the folder owned by git user and not by root user, because when you first copy it, it is automatically owned by root user.
4. Do the same for S3\_debug folder.
5. Note that S3 and S3\_debug is the new way of doing things. They replace minio and minio provisioner which are being deprecated.
6. Cd back to /appname. Take a look at the kustomization.yaml into the base directory, make the necessary modification. Cd to base/resources and take a look at the namespace.yaml and the auth-policy-patch.yaml and net-policy-patch.yaml . cd to appname/overlays/prod and take a look at the kustomization.yaml there to validate the paths (../../base) works.
7. Cd to prod/appname/ and vim the virtual-service.yaml. and modify it accordingly (add the app URL at “hosts:”, change the “gateways: -istio…xxxx to istio….public”. also look at the service entry yaml file. Then cd back to /appname/ and cat the kustomization.yaml there and verify the paths work.
8. Cd back to appname/overlays/prod/secrets. Go through every folder, look at their kustomizations files and the files mentioned into the resource section and perform the necessary updates. Add the secret and credential values and encrypt the files as needed with the “sops” command.
9. Cd back to /sipr-mission-manifests/mission/manifests/appname. In there do a “tree” to see if the file structure is what you desire.
10. Copy recursively /integrations/appname from maintuser to /home/git/sipr-mission-bootstrap/integrations/ then make the newly copied appname directory owned by git (chown -R git:git to /home/git/sipr-mission-bootstrap/integrations/appname)
11. Copy recursively mission-manifests/appname from maintuser to /home/git/sipr-mission-bootstrap/mission-manifests/ and make it owned by git like in step 10.
12. Cd to integrations/sewol/overlays/prod/secrets from there, cd into the creds repositories and modify or fill out the creds files AND ENCRYPT it with the “sops” command (sops -e <file-name.env> **>** <file-name.enc.env>)

* Appname-db-creds.env -------- somethingsupersecret
* Minio creds ( accesskey--- appname-minio-app-user, secretkey--- (openssl rand -hex 5) twice, apps3bucketuuid--- value from openssl, s3\_region---us-gov-west-1)
* Also check all the kustomization.yaml files to make sure they are properly configured.

1. Run a kustomize build to make sure it builds properly.
2. Now cd to sipr-mission-bootstrap/ and modify the values.yaml file to incorporate the new app.

* Try to keep it in alphabetical order. Add lines on “wave:1, wave:5 and their paths” to reflex your app. Refer to and Keep the same format as the apps already listed.

**Synker box settings**

1. Assume maintuser role. Then cd to apps-synker/cvs/ in there create a file for the app (cp jfac.csv sewol.csv, for example will create a duplicate of jfac and name it sewol). Then vim and modify the newly created file. This file should contain all the image tags and their registry paths.
2. Vim the .app-synker-update\_v1.15.0.sh script and modify it accordingly.

* Scroll down and add a line for the app under the appropriate group (IL2, IL4, IL5, MISC)
* Scroll down to the “case $choice in” section and add a line for the app.
* Scroll down to the “begin main” section and add you appname into the “APP\_ARRAY”. Keep it in alphabetical order.
* Run the script and build and push the images.

To get pass the SSL certificate error for git run this command : git config --local http.sslVerify false

Do not forget about the finalizer thing in argoCd ( in the sipr mission project) when you are trying to delete an application.