THIS WAS A SPECIFIC CUSTOMER EXAMPLE. CHANGE TO GENERAL DEPLOYMENT GUIDANCE!!!!

Deployment Automation

Support the orchestration of coordinated deployment activities. The deployment can be chained with build activities, but as we move to a build once and deploy everywhere model it is key to separate the two to ensure that deployments can be targeted to a specific environment with a specific build.

The deployment process is dependent on both the CI and Asset Management to work correctly. The expectation is that during deployment we are given a desired application or component, desired version, and target environment to deploy to.

Production like Deployment

The goal here is to use the same process for deploying to lower environments to how we deploy to production. This allows for testing of changes to both the process and code base as a unit, building confidence that our deployments will be successful too.

This requires some refactoring of the lower environment deployments as well as minor refactoring of production-like deployments to find a common middle ground.

To accomplish the model in use for production assumes that the files are locally available in an archive that can be extracted to the filesystem. Then use `pm2` to stop the existing app, re-link the new version to current and start the app with `pm2` again. So this is the model we want to use moving forward.

Some changes being made:

1. Tagging of source and packaging of deployable artifacts will be done by the CI system and published to Nexus as the source of truth.
2. Now the only thing we need to provide is the application name and version and we are all talking about the same thing (in source, CI, Nexus, and into deployment)
3. The deployment scripts will now live with the source code and not on the target machines to avoid them drifting apart in the future. There is a new folder in OnlineBind named deploy that contains and updated version of the old production deployment script.
4. This means that lower environment deployments will be able to use this method for deployment as well.
5. Deployments will be triggered from Jenkins in lower environments

Updated deployment scripts

The deployment scripts themselves are now located in the source directories. See the `blank` folder, specifcally the script, this is a refactored version of the original script from Ops. The new script does not live in isolation on the server, but is part of the deployable package and provides an additional opportunity to use the same fixed deployment process for QA, UAT, Pre-Stage, and Production to help improve the quality of deployment consistency as well.

In order to make this possible, the lower environments of QA and UAT have been modified to run as the production-like environments run. That is to use the `nodejs` user as opposed to `root` and run from the fixed location of `/data/nodejs/<appname>` . This process once implemented simply requires that users that must login be sure to change to the `nodejs` user before doing any pm2 commands.

Additionally this means that all environment specific deployments in the `ecosystem.json` file will no longer be used, and can be removed to avoid supporting conflicting and unused code.

Supporting deployment scripts

In addition to the actual deployment, the deployment automation jobs in Jenkins do a few additional tasks to support keeping the environments clean and avoid constantly re-deploying existing versions.

First is a cleanup script that will only keep the currently deployed version on the machine. This is less risk for a non-production environment as we can just request the last version be deployed or more likely if something is broken, fix it, and deploy the next version.