Developer Onboarding

Windows WSL

IDEA - WSL.docx (contributor: jdcl@amazon.com)

MacOS

Homebrew

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

AWS CLIv2

```
brew install awscli
```

Configure AWS CLI

Refer to instructions in Configure AWS CLI v2

pyenv

Install pyenv to simplify managing multiple python versions on your system.

```
brew install pyenv
```

Python v3.9.16

```
pyenv install --skip-existing 3.9.16
```

NVM (Node Version Manager)

Install nvm to simplify managing multiple Node versions.

```
brew install nvm
```

NodeJS v16.10.0

```
nvm install 16.10.0
nvm use 16.10.0

# to set default nodejs version to 16.10.0, run:
nvm alias default 16.10.0
```

AWS CDK v2.69.0

Note: Do **NOT** install CDK globally using `npm -g` or `yarn global add`

Follow the instructions below:

```
mkdir -p ~/.idea/lib/idea-cdk && pushd ~/.idea/lib/idea-cdk
npm init --force --yes
npm install aws-cdk@2.69.0 --save
popd
```

Upgrade Info

If you want to **upgrade** CDK version for your existing IDEA dev environment, run:

```
invoke devtool.upgrade-cdk
```

Yarn

```
brew install yarn
```

Docker Desktop (Optional)

Follow instructions on the below link to install Docker Desktop. (Required if you are working with creating Docker Images)

https://docs.docker.com/desktop/mac/install/

Amazon Linux 2

pyenv

```
curl https://pyenv.run | bash
export PYENV_ROOT="$HOME/.pyenv"
command -v pyenv >/dev/null || export PATH="$PYENV_ROOT/bin:$PATH"
```

Development libraries

```
sudo yum install -y @development zlib-devel bzip2 bzip2-devel readline-devel sqlite \ sqlite-devel openssl-
devel xz xz-devel libffi-devel findutils openldap-devel
```

Python v3.9.16

```
pyenv install --skip-existing 3.9.16
```

NVM (Node Version Manager)

Install nvm to simplify managing multiple Node versions.

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
```

NodeJS v16.10.0

```
nvm install 16.10.0
nvm use 16.10.0

# to set default nodejs version to 16.10.0, run:
nvm alias default 16.10.0
```

AWS CDK v2.69.0

Note: Do **NOT** install CDK globally using `npm -g` or `yarn global add`

Follow the instructions below:

```
mkdir -p ~/.idea/lib/idea-cdk && pushd ~/.idea/lib/idea-cdk
npm init --force --yes
npm install aws-cdk@2.69.0 --save
popd
```

Yarn

```
npm install yarn
```

Clone Git Repo

All CRs will be accepted only against **feature/idea-development**. Refer to Release Engineering for more details on branching mechanisms.

If you do no wish to make any changes or publish CRs, you can checkout the **feature/idea-mainline** branch.

```
git clone ssh://git.amazon.com/pkg/Solution-for-scale-out-computing-on-aws
cd Solution-for-scale-out-computing-on-aws
git checkout feature/idea-development
```

Virtual Environment

Note: If you are planning to use an IDE such as PyCharm, skip all of the below steps and continue with instructions outlined in: Setup PyCharm for Development

```
PYENV_VERSION=3.9.16 python -m venv venv
```

If the above PYENV_VERSION command is not working for any reason, you can create venv using below command:

```
~/.pyenv/versions/3.9.16/bin/python3 -m venv venv
```

Activate the virtual environment

```
source venv/bin/activate
```

Install Dev Requirements

```
pip install -r requirements/dev.txt
```

BigSur Note: cryptography and orjson library requirements fail to install on MacOS BigSur.

To fix **orjson**, run:

```
brew install rust
# Upgrade your pip
python3 -m pip install --upgrade pip
```

To fix **cryptography**, follow the instructions mentioned here:
https://stackoverflow.com/questions/64919326/pip-error-installing-cryptography-on-big-sur

```
env LDFLAGS="-L$(brew --prefix openssl@1.1)/lib" CFLAGS="-I$(brew --prefix openssl@1.1)/include" pip install
cryptography==36.0.1
```

Verify Dev Setup

Run below command to check if development environment is working as expected, run:

```
invoke -l
```

Running this command should print output like below:

```
Available tasks:

  admin.main (admin)                  call administrator app main
  build.administrator                 build administrator
  build.all (build)                   build all
  build.cluster-manager               build cluster manager
  build.data-model                    build data-model
  build.dcv-connection-gateway        build dcv connection gateway
  build.scheduler                     build scheduler
  build.sdk                           build sdk
  build.virtual-desktop-controller    build virtual desktop controller
  clean.administrator                 clean administrator
  clean.all (clean)                   clean all components
  clean.cluster-manager               clean cluster manager
  clean.data-model                    clean data-model
  clean.dcv-connection-gateway        clean dcv connection gateway
  clean.scheduler                     clean scheduler
  clean.sdk                           clean sdk
  clean.virtual-desktop-controller    clean virtual desktop controller
  devtool.build                       wrapper utility for invoke clean.<module> build.<module> package.<module>
  devtool.configure                   configure devtool
  devtool.ssh                         ssh into the workstation
  devtool.sync                        rsync local sources with remote development server
  devtool.upload-packages             upload packages
  docker.build                        build administrator docker image
  docker.prepare-artifacts            copy administrator docker image artifacts to deployment dir
  docker.print-commands               print docker push commands for ECR
  package.administrator               package administrator
  package.all (package)               package all components
  package.cluster-manager             package cluster manager
  package.dcv-connection-gateway      package dcv connection gateway
  package.scheduler                   package scheduler
  package.virtual-desktop-controller  package virtual desktop controller
  release.prepare-opensource-package
  release.update-version
  req.install                         Install python requirements
  req.update                          Update python requirements using pip-compile.
  scheduler.cli (scheduler)           call scheduler cli
  tests.run-integration-tests         Run Integration Tests
  tests.run-unit-tests                Run Unit Tests
  web-portal.serve                    serve web-portal frontend app in web-browser
  web-portal.typings                  convert idea python models to typescript
```

Clean, Build and Package

```
invoke clean build package
```

Run idea-admin.sh in Developer Mode

The **IDEA_DEV_MODE** environment variable is used to indicate if *idea-admin.sh* or *idea-admin-windows.ps1* should use the Docker Image or Run from sources.

- If IDEA_DEV_MODE=true, idea-admin.sh will execute administrator app directly using sources.
- If IDEA_DEV_MODE=false (default), idea-admin.sh will attempt to download the docker image for the latest release version and execute administrator app using Docker Container.

Export IDEA_DEV_MODE=true on your terminal, before executing idea-admin.sh on from project root.

Eg.

```
export IDEA_DEV_MODE=true
```

You will need to run **export IDEA_DEV_MODE=true**, each time you open a new Terminal session.

Adding IDEA_DEV_MODE to .zshrc or .bashrc is not recommended as you will not be able to seamlessly switch between dev mode vs non-dev mode and test the Docker Container based idea-administrator flow.

Pro Developer Tip

IntelliJ IDEA or PyCharm Users

If you are using IntelliJ IDEA or PyCharm, you can set default environment variables for terminals in your IDE.

- Navigate to Preferences Tools Terminal
- Add IDEA_DEV_MODE=true as one of the environment variables.

Any new terminal sessions from within the IDE will automatically include IDEA_DEV_MODE=true.

VS Code Users

If you are using VS Code, you can create a Terminal Profile in your settings.json file to have an easy way to spawn both Terminals. An example:

- Open your settings.json file in VScode (Apple Key + ,) - click one of the 'Edit in settings.json' links
- Look for your platform terminal settings - In this example we add a new profile on OSX -

VS Code - Terminal Settings for IDEA_DEV_MODE

```
    "terminal.integrated.profiles.osx": {
        "IDEA_DEV_MODE": {
            "overrideName": true,
            "path": "/bin/bash",
            "args": [
                "-l"
            ],
            "env": {
                "IDEA_DEV_MODE": "true"
            }
        }
```

Make sure to preserve the JSON syntax of your settings.json file or you will have problems!

Verify if Developer Mode is enabled

To verify, if Developer Mode is enabled, run below command. This should print **(Developer Mode)** at the end of the banner.

```
./idea-admin.sh about

 '####:'########::'########::::'###::::
 . ##:: ##.... ##: ##.....::::'## ##:::
 : ##:: ##:::: ##: ######:::'##:::. ##:
 : ##:: ##:::: ##: ##...:::: #########:
 '####: ########:: ########: ##:::: ##:

 Integrated Digital Engineering on AWS
          Version 3.0.0-beta.1
            (Developer Mode)
```

Suggested Development Settings

During the development phase it may be helpful to keep some of these settings in mind

1. Suspend ASG actions if you are working on an ASG-managed instance and intend to patch/update or are working on something where the process may crash. This can reduce the re-warm time for an instance.
2. Use the `idea-admin.sh` patch functionality if applicable for your updates.
3. Make sure to use the `debug` log profile settings.

4. Enable payload tracing for API requests to get more details. **Example**:

```
./idea-admin.sh config set Key=cluster.logging.audit_logs.enable_payload_tracing,Type=bool,Value=True --
aws-region us-east-1 --cluster-name idea-eada
```

5. Tho shalt always remember to rebuild packages (`invoke clean build package`)

6. Rebuilding the package can also take a target to speed up the process and only rebuild that specific page. For example - if you are working heavily on the cluster-manager - You can do something like `invoke clean.cluster-manager build.cluster-manager package.cluster-manager` . **NOTE**: Just be aware of any cross dependencies or changes to `data-model` and `sdk` ! If in doubt - clean/build/package them all. Better to make sure things are rebuilt and not rebuild incorrectly.