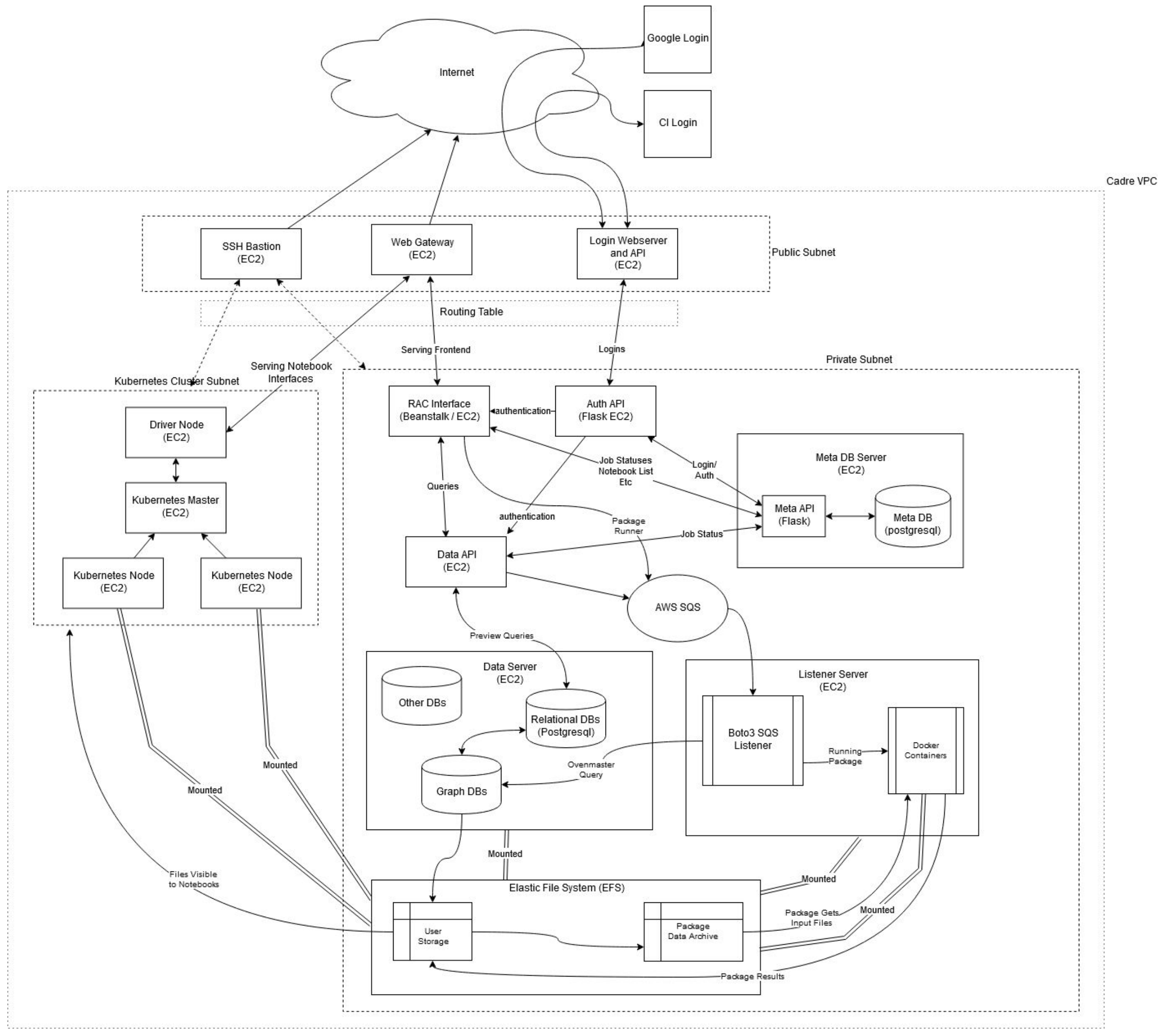
Cadre Architecture



Cadre backend components

Cadre-federated-login system

- Github : https://github.com/iuni-cadre/cadre-login
- Both production and dev versions deployed from master
- Use AWS cognito for federation
    - Identity providers : CiLogon, Google
    - Client ids can be found in cognito console when select correct region
    - CILogon is added as third party OIDC
    - For CILogon, you need to add custom attributes in order to available them in the response
    - AWS manages the domain for login as well
        - login.cadre.iu.edu - dev version
        - login-1.cadre.iu.edu - production version
- Service restart
    - Managed by supervisor process
    - Logs : /home/ubuntu/cadre-login/cadre_logging.log
    - Config : /home/ubuntu/cadre-login/conf/cadre.config
    - Important configs :

        [DEFAULT]
        - cadre_dashboard=https://cadre.iu.edu/gateway/?username=

        [DATABASE_INFO] - Related to metadb
        - database-host=
        - database-port=
        - database-name=
        - database-username=
        - database-password=

        [JUPYTERHUB]
        - jupyterhub-apihost=http://a2ac434e8f73511e9804d0a1687f12bb-955710421.us-east-1.elb.amazonaws.com/jupyter/hub/api/

        [AWS]
        - aws-access-key-id=
        - aws-secret-access-key=
        - region-name=
        - client-id=
        - redirect-uri=https://cadre.iu.edu/api/auth/cognito/callback
        - logout-redirect-uri=https://cadre.iu.edu

        - token-endpoint=https://login-1.cadre.iu.edu/oauth2/token
        - userinfo-endpoint=https://login-1.cadre.iu.edu/oauth2/userInfo

Cadre-data-api

- Github : https://github.com/iuni-cadre/cadre-data
- Both production and dev versions deployed from master
- Service restart
    - Managed by supervisor process
    - Make sure data api ec2 instance can create connections to database machines (mag postgres + neo4j, wos postgres + neo4j)
    - Logs : /home/ubuntu/cadre-data/cadre_data.log
    - Config : /home/ubuntu/cadre-data/conf/cadre.config
    - For the dev version, we use two aws sqs queues for now. One for janus graph and one for WOS
    - Important configs :

      ```
       [WOS_DATABASE_INFO]
      database-host=10.0.1.134
      database-port=5432
      database-name=wos
      database-username=
      database-password=

      [CADRE_META_DATABASE_INFO]
      database-host=10.0.1.192
      database-port=5432
      database-name=
      database-username=
      database-password=

      [MAG_DATABASE_INFO]
      database-host=10.0.1.197
      database-port=5432
      database-name=mag
      database-username=
      database-password=

      [MAG_GRAPH_DB_INFO]
      database-url=bolt://10.0.1.176:7687
      database-username=
      database-password=
      ```

```
[CADRE]
This is token endpoint from cadre-login repository
token-api=https://cadre.iu.edu/api/auth/authenticate-token

[AWS]
aws_access_key_id=
aws_secret_access_key=
region_name=us-east-1
queue_url=https://sqs.us-east-1.amazonaws.com/799597216943/cadre-job-listne
r-vpceast1.fifo
```

Cadre meta database

- Github : https://github.com/iuni-cadre/cadre-metadatabase
- Need to run only once, if you create a new ec2 instance for meta database

Cadre listeners

- ### Job Listeners
  - Job listeners are vastly different in production and dev versions
  - Production
    - Listeners running on jupyter hub driver instance
      - Login : SSH to production bastion
      - Run ssh_to_jupyter
    - Github: (Not from master) https://github.com/iuni-cadre/cadre-job-listener/tree/kub_run_packages
    - Job queue : https://sqs.us-east-1.amazonaws.com/799597216943/cadre-job-listner-vpceast1.fifo
    - 16 job listeners are running via supervisor
    - All the logs are in /home/ubuntu/cadre-job-listener/cadre_tool_listener.log
    - If you do changes to the python code, you need to restart supervisor processes
    - Make sure database connections are successful before starting the processes
    - EFS needs to be mounted
    - Job results are on /home/ubuntu/efs/home/cadre-query-results/{username in base32}/query-results
  - Dev

        - WOS
            - Deployed from master, in jupyter driver ec2
            - Login : ssh to dev bastion and run ssh_to_jupyter
            - Github: https://github.com/iuni-cadre/cadre-job-listener/tree/master
        - MAG Janus
            - Github: https://github.com/iuni-cadre/janus-job-listener/tree/mag-janus
            - Deployed in Janus server ec2 instance from mag-janus branch
            - Login : ssh to dev bastion and run ssh_to_janus_server_2_1
            - Make sure EFS is mounted
            - Java application
            - Install Java 8, apache maven
            - Build :
                - cd /home/ubuntu/janus-job-listener
                - mvn clean install
            - Deploy
                - Once the build successful, copy generated jars from target dir
                - cp target/janus-job-listener-0.0.1-SNAPSHOT* ~/lib/
                - Kill the running nohup process and start the new process
                    - cd /home/ubuntu/janus-job-listener/bin
                    - nohup sh start.sh > nohup.out &
                - In the future, we should start this as supervisord process
            - Logs : /home/ubuntu/janus-job-listener/cadre-janus-job-listener.log
            - Config : /home/ubuntu/lib/cadre_config.properties

- Tool creation Listener
    - This is called when user clicks create tool button from Marketplace
    - Tool info submitted to tool queue from cadre-interface
        - Queue urls :
          https://sqs.us-east-1.amazonaws.com/799597216943/cadre-tool-listner-vpceast1.fifo (production)

          https://sqs.us-east-2.amazonaws.com/799597216943/cadre-tool-queue.fifo (dev)

    - Create a docker image and upload to docker hub
    - https://github.com/iuni-cadre/cadre-wiki/wiki/Run-packages-with-Kubernetes
    - Run on jupyter driver ec2 instance
    - Make sure kubernetes, docker installed

    - Make sure you can pull images from cadre docker repository without prompting to get username and password. Follow the link in the wiki document.
    - Logs : /home/ubuntu/cadre-job-listener/cadre_tool_listener.log
- ### Package run Listener
    - This is called when user clicks on run package
    - Package creation does not associate with the listener. It only adds database entries to the meta database
    - Package run information submitted to the queue from cadre-interface middleware
    - Queue urls
        - https://sqs.us-east-1.amazonaws.com/799597216943/cadre-package-listner-vpceast1.fifo (production)
        - https://sqs.us-east-2.amazonaws.com/799597216943/cadre-package.fifo (dev)
    - When submit package run, we first get the tool id for the package from metadb, and pull the docker image with tool_id as the tag from dockerhub, cadre repository
    - Package run as a pod in kubernetes, you can run kubernetes commands to see whether the package run successfully
        - kubectl get pods -n jhub
        - kubectl logs -n jhub <pod_id>
    - Pods with completed or failed will be deleted when the next package run
    - Logs : /home/ubuntu/cadre-job-listener/cadre_package_listener.log
- Configs
    - Except for Janus job listener, for all the other listeners, config file located in /home/ubuntu/cadre-job-listener/conf/cadre.config
    - It contains configs related to all the listeners

Notebooks

- For notebooks, we use jupyterhub
- Installation and other userful info can be found in https://zero-to-jupyterhub.readthedocs.io/en/latest/amazon/step-zero-aws.html
- About configs and instructions : https://github.com/iuni-cadre/cadre-notebooks
- We use jupyter REST API to generate tokens and create notebook servers for users https://jupyterhub.readthedocs.io/en/stable/reference/rest.html
- Admin username is IUNITester, if you need an admin token, you need to navigate to the admin panel and create an api token. We have created admin tokens and they are used in cadre-interface middleware. If you revoke api tokens, you need to redeploy the Elastic beanstalk instance with a new api key.
- Config.yml can be found in the jupyter driver home directory.

- To redeploy jupyterhub
    - helm upgrade -f config.yaml jhub jupyterhub/jupyterhub --version=0.8.2

Kubernetes cluster

- Both production and dev vpcs, contains kubernetes cluster with 1 master, 4 nodes and one driver node
- Production cluster configured to auto scale according to the load, dev is not yet configured.
- To run all the kubernetes operations, you need to login to driver node. (using ssh_to_jupyter from bastion)
- kops commands use to update the cluster, node configurations etc

Janus Graph

- Janus cluster only available in ohio vpc
- Running cluster includes
    - 1 jansu server
    - 1 elastic search server
    - 3 cassandra nodes (1 seed + 2 nodes)
- Start janus server
    - Ssh to janus server instance
    - Kill the running nohup process
    - cd janusgraph, run nohup bin/gremlin-server.sh conf/gremlin-server/socket-http-gremlin-server.yaml > janus.out &

Cadre Datasets

- Separate EC2 instances for MAG and WOS
- Each ec2 instance contains postgres DB and neo4j DB
- Each should have EFS mounted

AWS operations

- VPCs
    - Design : https://github.com/iuni-cadre/cadre-wiki/wiki/Create-new-VPC-for-Cadre
    - 2 VPCs
    - Dev - Ohio region

    - Production - Virginia region
- Security groups
    - 2 main security groups defined for each vpc
    - cadre-vpc-pub-sg - for public facing ec2 instances
    - cadre-backend-sg - for private ec2 instances
    - Always try to use one of the above security groups if you want to add new ec2 instances