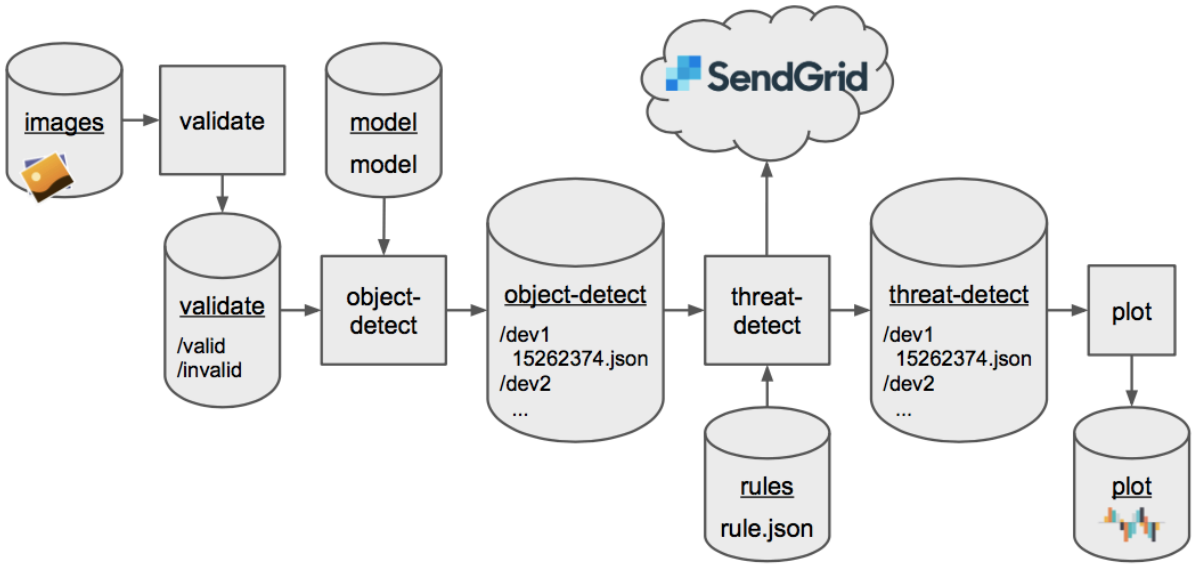
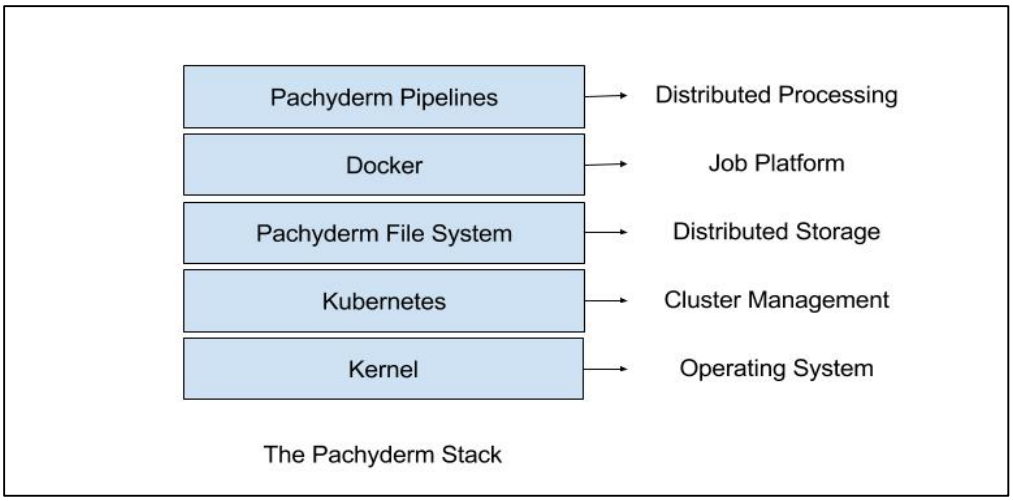
Putting it all together: Pipeline

So far, we had decided to implement a system which looks like this



The images for threat detection will be stored in a repository. The images will go through 'validate' process which will check images based on resolution and other parameters, and create file names. The 'validate' 'image' in the Docker container will check for validity of image. It will store the valid and invalid images separately. The TensorFlow model developed for object detection for different clients will be stored in 'model' repository. The Python 'image' will consist of codes that are executable using Python libraries in the 'TensorFlow' image in Docker container. The 'object detect' will process images and detect 'objects' in the image. The 'objects' detected in the image will be classified as threat or friendly based on rules in the file 'rule.json'. For the purpose of demonstration, we will consider only one rule. Different clients can have different threat perception and this will help determine if the 'object' in image is a threat or not. 'Threat-detect' 'image' in the Docker container will contain the package necessary to run for threat detection. The threat will be described in form of a json file with the date and time the threat was detected along with its location. This information will go to the user using an email service such as SendGrid. The threat once detected will be stored in the separate repository for regulatory purpose. This storage will happen in a 'static' 'object' storage platform like S3. For internal customer analytics, we will plot the number of threats against user or location. All the computation will happen on EC2 instance. Kubernetes will manage the EC2 instances (VMs). We will create a Pachyderm 'image' stored in Docker container called as 'pachd', that will automate the entire process of running different 'images'. It will under the hood manage Kubernetes and optimize the utilization of EC2 instance.



The image shows how we incorporate open-source software, working on cloud, managing and scheduling the processes using Pachyderm pipelines.

Implementation:

First, we login to a virtual machine which has Pachyderm, TensorFlow and Kubernetes installed on it. I executed **ssh pachrat@XXX.XXX.XXX.XXX** on my Mac Terminal. Once we login, we run the following commands.

kubectl get all – command that shows all that is running on Kubernetes.

```
NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/dash    1         1         1            1           1h
deploy/etcd    1         1         1            1           1h
deploy/pachd   1         1         1            1           1h

NAME                  DESIRED   CURRENT   READY     AGE
rs/dash-6c9dc97d9c    1         1         1         1h
rs/etcd-7dbb489f44    1         1         1         1h
rs/pachd-75c6c9b79f   1         1         1         1h

NAME                         READY     STATUS    RESTARTS   AGE
po/dash-6c9dc97d9c-bz4kk     2/2       Running   0          1h
po/etcd-7dbb489f44-625d5     1/1       Running   0          1h
po/pachd-75c6c9b79f-rvqfq    1/1       Running   0          1h

NAME                                TYPE                    CLUSTER-IP              EXTERNAL-
IP   PORT(S)                                                    AGE
svc/dash          NodePort    10.105.88.60    <none>        8080:30080/TCP,8081:30081/TCP
                        1h
svc/etcd          NodePort    10.105.9.158    <none>        2379:32379/TCP
                        1h
svc/kubernetes    ClusterIP   10.96.0.1       <none>        443/TCP
                        1h
svc/pachd         NodePort    10.106.62.174   <none>        650:30650/TCP,651:30651/TCP,652:
30652/TCP,999:30999/TCP   1h
```

We see that 3 images are running. ‘dash’ will create a dashboard. ‘etcd’ tracks what data has been processed or any more data needs to be processed etc.

kubectl get pods - command that shows the status of pods

```
NAME                     READY     STATUS    RESTARTS   AGE
dash-6c9dc97d9c-bz4kk    2/2       Running   0          1h
etcd-7dbb489f44-625d5    1/1       Running   0          1h
pachd-75c6c9b79f-rvqfq   1/1       Running   0          1h
```

pachctl version - command that shows the version of Pachyderm installed

```
COMPONENT           VERSION
pachctl             1.6.5
pachd               1.6.5
```

git clone https://github.com/dwhitena/mgmt690-pipeline.git- command that clones folder named mgmt690-pipeline from github.

```
Cloning into 'mgmt690-pipeline'...
remote: Counting objects: 71, done.
remote: Compressing objects: 100% (51/51), done.
remote: Total 71 (delta 21), reused 58 (delta 14), pack-reused 0
Unpacking objects: 100% (71/71), done.
Checking connectivity... done.
```

Checking the mgmt690-pipeline folder shows the images in that folder

```
pachrat@mgmt6:~/mgmt690-pipeline$ ls
object-detect       plot       README.md    threat-detect       validate
object-detect.json  plot.json  test_images  threat-detect.json  validate.json
```

This shows the object-detect, threat-detect, plot and validate folder with the json files in it. The test images folder will contain the images on which we will perform threat detection.

pachctl create-repo rules
pachctl list-repo

```
NAME                CREATED              SIZE
rules               5 seconds ago        0B
model               12 seconds ago       0B
images              About a minute ago   0B
```

The command above will create a repository for rules. List the repo to display the 3 repository.

wget http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v1_coco_11_06_2017.tar.gz
--2017-11-28 19:18:04-- http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v1_coco_11_06_2017.tar.gz

```
Resolving download.tensorflow.org (download.tensorflow.org)... 2607:f8b0:4006:813::2010, 172.217.12.144
Connecting to download.tensorflow.org (download.tensorflow.org)|2607:f8b0:4006:813::2010|:80... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 128048406 (122M) [application/x-tar]
Saving to: ‘ssd_mobilenet_v1_coco_11_06_2017.tar.gz’

ssd_mobilenet_v1_co 100%[===================>] 122.12M   326MB/s    in 0.4s

2017-11-28   19:18:05   (326   MB/s)   -   ‘ssd_mobilenet_v1_coco_11_06_2017.tar.gz’   saved
[128048406/128048406]
```

tar -xvf ssd_mobilenet_v1_coco_11_06_2017.tar.gz

```
ssd_mobilenet_v1_coco_11_06_2017/
ssd_mobilenet_v1_coco_11_06_2017/model.ckpt.index
ssd_mobilenet_v1_coco_11_06_2017/model.ckpt.meta
ssd_mobilenet_v1_coco_11_06_2017/frozen_inference_graph.pb
ssd_mobilenet_v1_coco_11_06_2017/model.ckpt.data-00000-of-00001
ssd_mobilenet_v1_coco_11_06_2017/graph.pbtxt
```

The commands above downloads TensorFlow and relevant libraries.

pachctl put-file model master -c -f frozen_inference_graph.pb – Command puts model file in model repository with the graph.

pachctl list-repo – Command that lists the Pachyderm repositories

```
NAME                CREATED             SIZE
model               3 minutes ago       27.83MiB
rules               3 minutes ago       0B
images              4 minutes ago       0B
```

To modify the rule.json file, we will first navigate to that folder and then add the email address to which alert emails are to be sent. The following commands are used to do that.

pico rule.json
cat rule.json

```
{
  "from_email": "support@baimsecurity.com",
  "to_email": "pgujela@purdue.edu",
  "threat_classes": [1, 3, 4, 8]
}
```

cat validate.json – Command to check the contents of validate file

```
{
  "pipeline": {
    "name": "validate"
  },
  "transform": {
    "image": "dwhitena/mgmt-validate",
    "cmd": [ "/bin/ash" ],
    "stdin": [
      "python /code/validate.py $images /pfs/out/valid /pfs/out/invalid"
    ]
  },
  "parallelism_spec": {
    "constant": "1"
```

```
  },
  "input": {
    "atom": {
      "repo": "images",
      "glob": "/*"
    }
  }
}
```

We can see from the codes that valid and invalid images are stored in different folders.

pachctl create-pipeline -f validate.json
pachctl create-pipeline -f object-detect.json
pachctl create-pipeline -f threat-detect.json

All the above commands will execute the respective files in Kubernetes by Pachyderm. First, we will validate the image, detect objects and then detect threats.

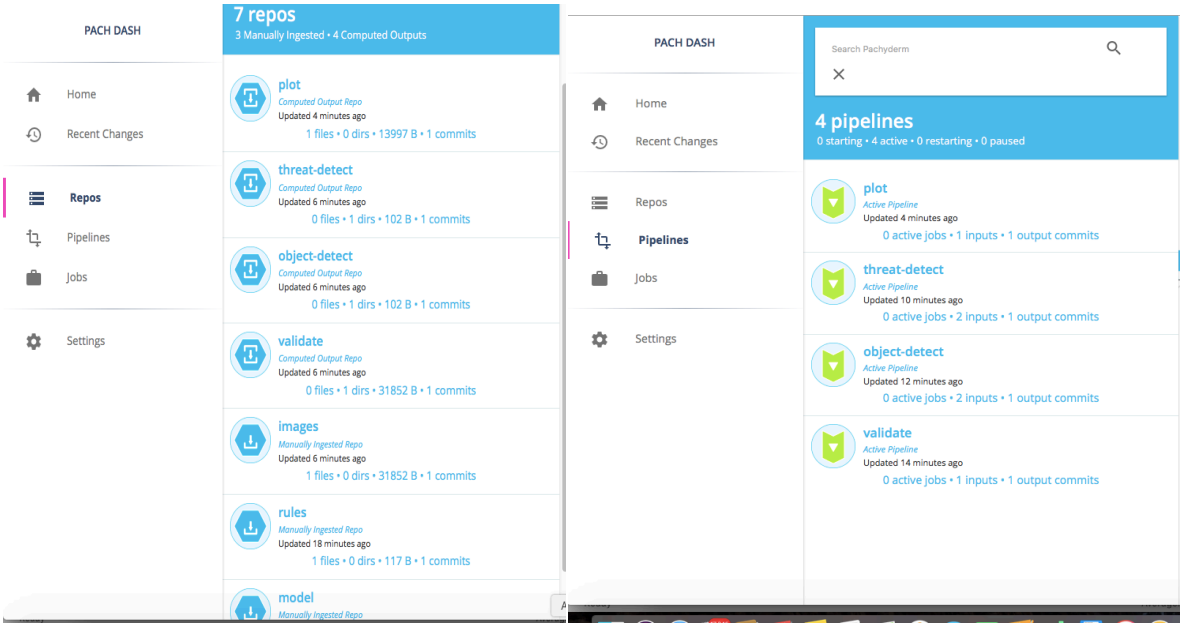
kubectl get pods – This command will display all the Docker images that Kubernetes is running

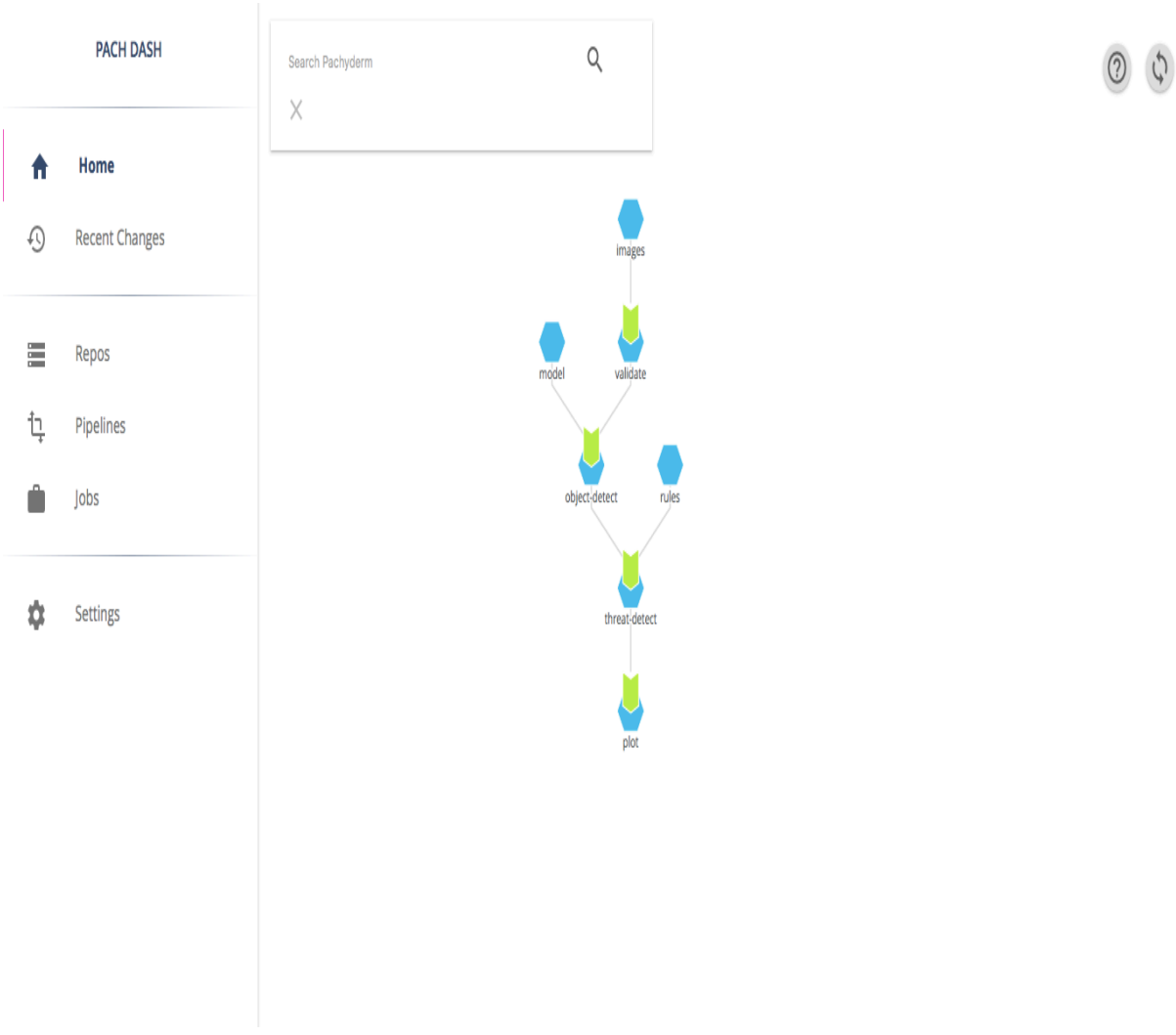
```
NAME                                READY     STATUS    RESTARTS   AGE
dash-6c9dc97d9c-bz4kk               2/2       Running   0          1h
etcd-7dbb489f44-625d5               1/1       Running   0          1h
pachd-75c6c9b79f-rvqfq              1/1       Running   0          1h
pipeline-object-detect-v1-9hgwm     2/2       Running   0          2m
pipeline-threat-detect-v1-d68kf     2/2       Running   0          41s
pipeline-validate-v1-z7fb9          2/2       Running   0          4m
```

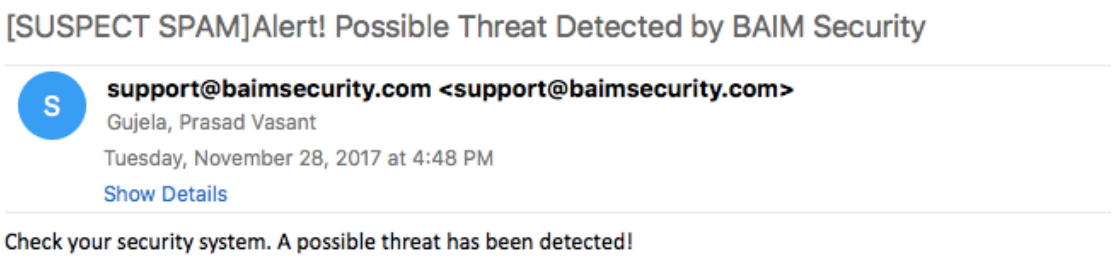
pachctl list-job – This command will list out the files that Pachyderm is running and its status

```
ID                                                                                  OUTPUT
COMMIT                                          STARTED        DURATION             RESTART
PROGRESS  DL       UL       STATE
a768938b-33fc-4d2d-922c-d2aeb111dcb1 plot/48c11ccad0c64d348e85e3e18bcae9b6        4 minutes
ago Less than a second 0       1 + 0 / 1 102B     13.67KiB success
57bd23eb-2484-4d5a-a2cf-b47cd20c3151    threat-detect/74fec69fa6cf472f9684c2dcb8aef916    7
minutes ago Less than a second 0       1 + 0 / 1 219B     102B     success
4d2706d3-1fb2-44d6-af11-f5392efbe72f    object-detect/11ae39f89a4c4d9ca5abd4cf365002f1    7
minutes ago 8 seconds          0       1 + 0 / 1 27.86MiB 102B     success
ac9bbddc-38a1-4a5f-a004-921a6d87cc3f validate/7cf96f8d97994a75a7faeeb2b8b3589e     7 minutes
ago Less than a second 0       1 + 0 / 1 31.11KiB 31.11KiB success
```

We can see that all files have ran successfully. We have even finished executing the plot command. We can see the files and repositories created in Pachyderm along with the graph.


PACH DASH
Home
Recent Changes
Repos
Pipelines
Jobs
Settings
7 repos
3 Manually Ingested • 4 Computed Outputs
plot
Computed Output Repo
Updated 4 minutes ago
1 files • 0 dirs • 13997 B • 1 commits
threat-detect
Computed Output Repo
Updated 6 minutes ago
0 files • 1 dirs • 102 B • 1 commits
object-detect
Computed Output Repo
Updated 6 minutes ago
0 files • 1 dirs • 102 B • 1 commits
validate
Computed Output Repo
Updated 6 minutes ago
0 files • 1 dirs • 31852 B • 1 commits
images
Manually Ingested Repo
Updated 6 minutes ago
1 files • 0 dirs • 31852 B • 1 commits
rules
Manually Ingested Repo
Updated 18 minutes ago
1 files • 0 dirs • 117 B • 1 commits
model
Manually Ingested Repo
PACH DASH
Home
Recent Changes
Repos
Pipelines
Jobs
Settings
Search Pachyderm
4 pipelines
0 starting • 4 active • 0 restarting • 0 paused
plot
Active Pipeline
Updated 4 minutes ago
0 active jobs • 1 inputs • 1 output commits
threat-detect
Active Pipeline
Updated 10 minutes ago
0 active jobs • 2 inputs • 1 output commits
object-detect
Active Pipeline
Updated 12 minutes ago
0 active jobs • 2 inputs • 1 output commits
validate
Active Pipeline
Updated 14 minutes ago
0 active jobs • 1 inputs • 1 output commits


PACH DASH
Home
Recent Changes
Repos
Pipelines
Jobs
Settings
Search Pachyderm
images
model
validate
object-detect
rules
threat-detect
plot



The image will show the alert mail received after processing the threat image from the company ‘BAIM Security’. In future, we can add image that was used to detect threat in the email. A better way of alerting would be through text or call on cell phone. We so far tested the architecture on single image. Next time, we will productionize it to run many images automatically.