OnServ On-Call Runbook

Component Guides

Table of Contents

Useful Dashboards

Traffic Management & Routing

- OnServ - Istio (Chronosphere) - ingress gateway metrics. Useful for seeing error rates on public / rpc gateways and various resource metrics for the gateways. Useful to compare cluster-level or DT-level traffic.
- Istio Control Plane Dashboard (Chronosphere) - istiod metrics. Control plane for istio which manages the gateway pods.
- ~~OnServ - Istio - ingress gateway metrics. Useful for seeing error rates on public / rpc gateways and various resource metrics for the gateways~~
- COSM:OS Istio - can see client side metrics (ie. request volume, success rates, etc) per service
- COSM:OS Istio Services

Celery + RabbitMQ

- Chronosphere: Celery Overview, dashboard for metrics about celery workers
- Chronosphere: RabbitMQ Overview, dashboard for metrics about K8s RabbitMQ clusters. Can specify individual EKS cluster. Includes message stats as well as resource metrics
- ~~RabbitMQ Overview - dashboard for metrics about K8s RabbitMQ clusters. Can specify individual EKS cluster. Includes message stats as well as resource metrics~~
- ~~RabbitMQ Overview Main - similar dashboard as above. Aggregates metrics for all brokers in app clusters together~~
- ~~RabbitMQ Overview (stage) - dashboard for stage~~
- Sally Celery Dashboard - for task application metrics ie. task timings, start delays etc
- Celery Start Delay - more start delay metrics by DT/namespace
-
- ~~Celery Tasks (stage) - dashboard for stage~~

Caching

- Redis Connection Counts - AWS us-east-1 prod-live Cloudwatch

Crypto

- Chronosphere: Crypto Service K8s
  - US stage
  - CA stage
  - US prod sandbox
  - CA prod sandbox
- Chronosphere: Crypto Service
  - US prod live
  - CA prod live

Useful Slack Channels

#prod-issues-eng - updates, alarms, and monitors to prod, manual changes are prefixed with a #prodops message that get

#stage-issues-eng - same thing has #prod-issues-eng but for stage. Prefixed with #stageops

#prod-incidents
#ask-online-services
#ask-kubernetes

#bot-onserv-alerts (private channel)
#bot-crypto-rollbar-prod (only for live)
#bot-crypto-rollbar-stage (only for live)

Traffic Management & Routing Runbook

Adding IP for Allowlisting

New Vendor Onboarding Backend Steps

New Vendor Onboarding Backend Steps (IT / INFRA)

Updating employee-ip-restricted.conf

Prerequisites:

- Need access to OneLoginAdmin

Process:

1. Add the new IP to https://github.com/Affirm/cosmops/blob/master/kubernetes/resources/src/istio/gateways/public/files/filter-client-ip/employee_ips.lua and add a relevant comment
2. Change the pod annotation in the public ingressgateway istiooperator resource to any other value in order to trigger an automatic rollout restart of the ingress gateway (so that it will pick up the configmap changes from above
    a. https://github.com/Affirm/cosmops/blob/d020e5e6acb4bd85324510737faaeeef16180a12/kubernetes/resources/src/istio/gateways/public/public-ingressgateway.yaml#L124
3. Run `make build_all` from within the `kubernetes` directory
4. Make a PR, get a review done, and merge
5. Make a new release tag and deploy to kubernetes clusters by following: https://confluence.team.affirm.com/display/INFRA/CosmOps+Development#CosmOpsDevelopment-Deployment
6. If access to S3-hosted apps is needed (e.g. affirm_design_system, axp, bank portal, crm, pricing portal), follow the following steps to update Cloudflare cdn1:
    a. Add IPs to the list in https://github.com/Affirm/terraffirm/blob/master/modules/cdn/cloudflare/waf/main.tf#L15. There is a list for the live env, and a list for the sandbox env (which includes all the live IPs and a few sandbox-specific ones).
    b. To apply the changes, it has to be done locally on the CLI because the Cloudflare WAF module doesn't support Spacelift at the moment. You have to log in to BOTH the prod US and prod CA accounts, then you can run "terragrunt apply" from the following directories:
        i. envs/affirmprod/us-east-1/prod-live/cdn/cloudflare/waf
        ii. envs/affirmprod/us-east-1/prod-sandbox/cdn/cloudflare/waf

    Note that these resources only exist in prod in terraform because they're defined at the account level in Cloudflare so they're used across stage and prod, but only tracked under prod in code.

Debugging 404s

Check where the error is coming from: usually istio or nginx

1. Check nginx access logs → if the requests in question are making it here then the problem is most likely nginx or the application
    a. check the application pods to see if the requests are making it to the application - if they are, then the issue is most likely the application
    b. otherwise, look at the nginx logs and see if you can find any useful information in there (ie. redirects etc)
2. Check istio ingress gateway logs → if the requests in questions are making it here but not nginx access logs, then the problem is most likely a misconfiguration in istio

The most common places that 404s stem from are:

1. **istio virtual services**
    a. if the route is not defined in the istio virtual service, the user will get a 404

i. check using istioctl. Replace the app label with the correct ingress gateway depending on if the service in question is public or rpc

```
istioctl proxy-config route $(kubectl -n istio-system get pods -l
app=rpc-ingressgateway -o name | head -n 1).istio-system
```

2. **nginx -** try kubectl exec-ing into the relevant pod and looking at the nginx configs in `/etc/nginx` for a good sense of what configurations might be related to the issue
   a. we protect internal (employee-only) URLs by applying an IP filter that returns a 404 if the ip is not approved
   b. public nginx configs (nginx.conf) have a location block(s). If the route does not match any of the location blocks in the nginx.conf, then a 404 will be returned.
   c. in most nginx configs we redirect 403s to 404s (which is desirable for security reasons)
   d. look at nginx access logs for any useful information about possible redirects etc.
   e. try grepping for the specific path that is error-ing to see if any special behavior/routing is happening for it and could be causing an issue

Istio Runbook

Change Canary Traffic Split

Use the Canary Ramp Buildkite job to change the canary percentage weight for a given DT (this will reweight for both web and rpc). This uses a script called vsctl to update VirtualServices. Note that canary weights are reset back to their default values (as committed to git) on each DT redeploy.

You can also manually update weights using kubectl in the event the reweight script doesn't work. Make sure the applicable canary pods exist first! The validation webhook should stop you from sending traffic to deployments with zero replicas though.

Example manual command with kubectl on axp-rpc and axp-public

```
Unset
kubectl --as=system:serviceaccount:default:cosmos-cluster-admin-privileged
--as-group=system:authenticated -n axp edit vs axp-rpc


Change the canary destination weight
...
 - route:
   - destination:
     host: axp-rpc.axp.svc.cluster.local
```

```
    port:
     number: 443
   weight: 100
  - destination:
    host: axp-rpc-canary.axp.svc.cluster.local
    port:
     number: 443
   headers:
    response:
     add:
      X-Affirm-Canary: "true"
   weight: 0


kubectl --as=system:serviceaccount:default:cosmos-cluster-admin-privileged
--as-group=system:authenticated -n axp edit vs axp-public

Change the canary destination weight
...
  route:
  - destination:
    host: axp-public.axp.svc.cluster.local
    port:
     number: 443
   weight: 100
  - destination:
    host: axp-public-canary.axp.svc.cluster.local
    port:
     number: 443
   headers:
    response:
     add:
      X-Affirm-Canary: "true"
   weight: 0
```

Updating mTLS cert on rpc ingress gateway

(rpc credential)
will manifest as an SSL error like:

```
Error: [('SSL routines', 'tls_process_server_certificate', 'certificate verify failed')]
```

This means the cert has expired and needs to be renewed and then updated on ingress gateway. Follow instructions here:

Doc with instructions to update cert + secret

If you want to confirm that the cert has expired, see section below.

Check when cert is valid

Get the cert from the K8s secret and decode it:

```
kubectl get secret rpc-credential -n istio-system -o
jsonpath="{.data['tls\.crt']}" | base64 --decode > current-tls.crt
```

Get start date and end date for cert:

```
openssl x509 -startdate -enddate -noout -in current-tls.crt
```

Vault Certificate Password/Credentials

As of March 2023, we handle all of our certs in Vault on EC2 machines. Credentials for unsealing vault are located in 1Password https://affirm.1password.com/vaults/details/6jjr4yk2r4avjqounigd7vbbsm. Please also page #ask-infrastructure-engineering in case there's an incident with certs and vault.

InfraEng's Vault Cert Renewal
https://confluence.team.affirm.com/display/INFRA/Vault+Cert+Renewal

Use istioctl in production (requires OneLoginAdmin)

This is dangerous as it bypasses the impersonation step which protects against accidental changes to production clusters. Use with caution!

This will not be necessary after https://github.com/Affirm/cosmops/pull/751 is merged
- Allowing usage of istioctl in prod without impersonation.

Because istioctl, unlike kubectl, does not contain a command line argument for impersonation a service account, it is necessary to make manual changes to your ~/.aws/credentials and ~/.kube/config files to allow the port forwarding necessary for istioctl to work in production.

For each cluster you want to run istioctl in (us-east-1-prod-live-main in this example):

1. Add the following new profile to to ~/.aws/credentials:

```
[us-east-1-prod-live-main-cluster-admin]
role_arn = arn:aws:iam::906324658258:role/eks-cluster-admin-prod-live-main
# Set as appropriate for the cluster
source_profile = affirm-prod  # Or affirm-prod-canada, etc.
```

2. In ~/.kube/config, look for the user for the cluster in question under the users top-level key, then change the value for AWS_PROFILE to the name of the new profile:

```
- name: arn:aws:eks:us-east-1:906324658258:cluster/prod-live-main
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      args:
      - --region
      - us-east-1
      - eks
      - get-token
      - --cluster-name
      - prod-live-main
      command: aws
      env:
      - name: AWS_PROFILE
        value: us-east-1-prod-live-main-cluster-admin  # Was affirm-prod
      provideClusterInfo: false
```

If you cannot find the cluster you're looking for, you need to use scc to log into the cluster at least once before it will appear in the kubeconfig.

3. Now all kubectl/istioctl commands will run as cluster admin. Use istioctl as desired.
4. Undo the change in step 2 to avoid running all future commands as cluster admin!

View Ingress Gateway Routes

For now, these won't work in prod unless you first follow the steps in **Use istioctl in production**

These commands dump the Envoy route configuration from a randomly selected ingress gateway pod. If you want to see the configuration for a specific pod, replace the $(...) command substitution with the pod name in question.

Public ingress gateway

```
$ istioctl proxy-config route
NAME                                                DOMAINS                    MATCH                    VIRTUAL SERVICE
https.443.https.public-gateway.istio-system         affirm.com, *.affirm.com   /*                       public-virtual-service.monolith
https.443.https.public-gateway.istio-system         affirm.com, *.affirm.com   /*                       public-virtual-service.monolith
https.443.https.public-gateway.istio-system         affirm.com, *.affirm.com   /api/v3/features/*       public-virtual-service.monolith
https.443.https.public-gateway.istio-system         affirm.com, *.affirm.com   /mordor/debugapp/*       public-virtual-service.monolith
https.443.https.public-gateway.istio-system         affirm.com, *.affirm.com   /internal/cms/*          public-virtual-service.monolith
...
```

RPC ingress gateway

```
$ istioctl proxy-config route $(kubectl -n istio-system get pods -l app=rpc-ingressgateway -o name | head -n 1).istio-system
NAME                                            DOMAINS                                                MATCH      VIRTUAL SERVICE
 https.443.https-l7.rpc-gateway.istio-system     alm.prod-live-bravo.prod.aff, alm.prod.aff             /*         alm-rpc-l7.alm
https.443.https-l7.rpc-gateway.istio-system     axp-rpc2-lv.prod-live-bravo.prod.aff, axp.prod.aff     /*         axp-rpc-l7.axp
```

```
...
```

View Ingress Gateway Envoy Configuration

Dumping the Envoy configuration for an ingress gateway is useful for confirming that the Envoy filter chain is configured as expected.

1. Run `istioctl dashboard envoy deployment/public-ingressgateway.istio-system`
    a. This sets up port forwarding to a randomly selected public ingress gateway (can be substituted with other ingress gateways as well) pod and automatically opens the Envoy dashboard in your browser.
2. Click on "config_dump" to see the full Envoy configuration in JSON format
    a. Or, to get nice syntax highlighted paged output in YAML format, run: `curl http://localhost:15000/config_dump | yq -PC | less -r`
    b. Searching in less doesn't work very well with colored output because the ANSI escape sequences get in the way at color boundaries. Drop the -C from the yq command to remove colors.
3. Search using ⌘F for "dynamic_listeners" to jump straight to the filter chain configuration

Restart istiod

Istiod is the control plane for istio. Sometimes it can run into certificate related issues that cause the ingress gateways to be unhealthy. In these cases, restarting it can help get it into a good state. Currently the istiod deployment is named istiod-1-16-2. Restart command:

```
kubectl -n istio-system rollout restart deploy/istiod-1-16-2
```

Make sure to use impersonation flags (`--as=system:serviceaccount:default:cosmos-cluster-admin-privileged --as-group=system:authenticated`) or `kudo` in prod

Change CPU/Memory of ingressgateways without code change

When you need to change the CPU/memory resources of an ingressgateway, edit the istiooperator and not the deployment directly, eg:

```
kubectl -n istio-system edit istiooperator rpc-ingressgateway
```

This is because the resources are set up the operator, any changes to the deployment itself will get overwritten

Useful Prometheus Queries

Istio metrics documentation: https://istio.io/latest/docs/reference/config/metrics/

Total requests per second by ingress gateway
sum(rate(istio_requests_total[5m])) by (source_workload)

Public ingress gateway web requests per second by service
sum(rate(istio_requests_total{source_workload="public-ingressgateway"}[5m])) by (destination_service)

Internal ingress gateway connections per second by service
sum(rate(istio_tcp_connections_opened_total{source_workload="internal-ingressgateway"}[5m])) by (destination_service)

RPC ingress gateway web requests per second by service
sum(rate(istio_requests_total{source_workload="rpc-ingressgateway"}[5m])) by (destination_service)

Monolith web response codes (0 equivalent to nginx 499)
sum(rate(istio_requests_total{source_workload="public-ingressgateway", destination_service="web.monolith.svc.cluster.local"}[5m])) by (response_code)

Public web 5xx percentage by service
sum(rate(istio_requests_total{source_workload="public-ingressgateway", response_code=~"5.*"}[15m])) by (destination_service) / sum(rate(istio_requests_total{source_workload="public-ingressgateway"}[15m])) by (destination_service)

RPC 5xx percentage by service
sum(rate(istio_requests_total{source_workload="rpc-ingressgateway", response_code=~"5.*"}[15m])) by (destination_service) / sum(rate(istio_requests_total{source_workload="rpc-ingressgateway"}[15m])) by (destination_service)

RPC ingress gateway CPU usage (% of request)
sum(rate(container_cpu_usage_seconds_total{namespace="istio-system", pod=~"rpc-.*"}[5m])) by (pod)/sum(kube_pod_container_resource_requests{namespace="istio-system", pod=~"rpc-.*", resource="cpu"}) by (pod)

RPC ingress gateway memory usage (% of request)
sum(container_memory_usage_bytes{namespace="istio-system", pod=~"rpc-.*"}) by (pod)/sum(kube_pod_container_resource_requests{namespace="istio-system", pod=~"rpc-.*",resource="memory"}) by (pod)

Deployment replicas in istio-system namespace
kube_deployment_spec_replicas{namespace="istio-system"}

Split/weigh traffic between clusters

Check if errors are coming from one specific cluster:
https://affirm.chronosphere.io/dashboards/d/kubernetes-cluster-comparison/kubernetes-cluster-comparison?orgId=1

Use this buildkite pipeline https://buildkite.com/affirm-inc/reweight-cluster-traffic

If you get errors with validation, set the following environment variable
“DO_NOT_VALIDATE_CLUSTER_WEIGHTS=1”

For more information, see Kubernetes (COSMOS) Runbook “Split/weigh traffic between clusters”

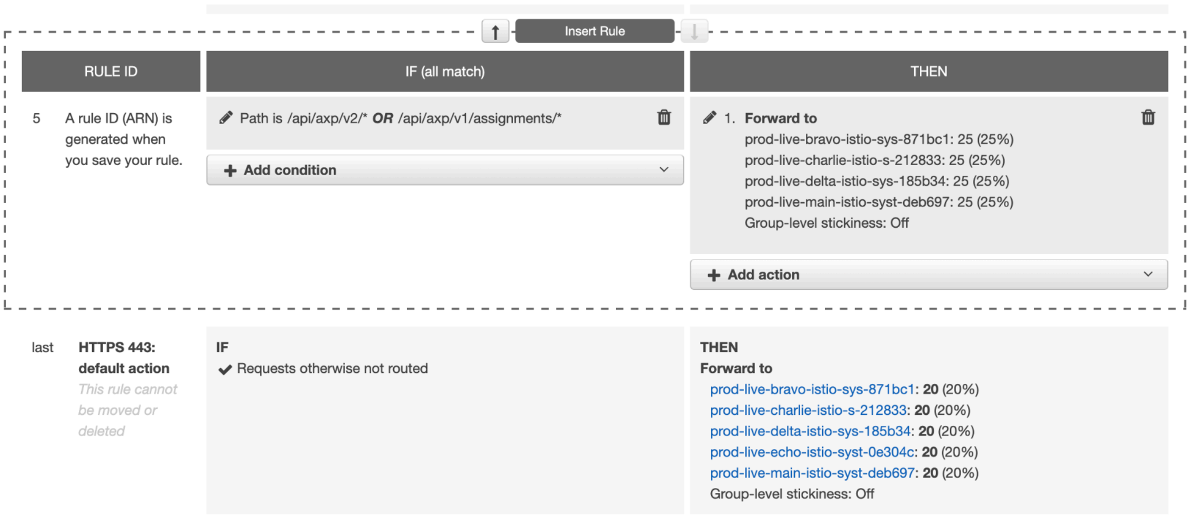
Splitting traffic between clusters in the AWS Console

Public Web
There are two load balancers (monolith and static-ip-prod).

RPC
RPC weights are controlled by DNS records. Route53 > Hosted Zones > prod.aff

Manually Weight Traffic Off a Cluster for a Public DT

1. Find the paths for the DT in question by looking at their public virtual service
   a. ie. for axp, the prefixes are /api/axp/v2/ and /api/axp/v1/assignments/
2. In the AWS console, add a new rule to the HTTPS:443 listener in the monolith alb
   a. insert the rule above the default routing rule
   b. example for axp, weighting traffic off of echo



RULE ID	IF (all match)	THEN
5 A rule ID (ARN) is generated when you save your rule.	Path is /api/axp/v2/* ***OR*** /api/axp/v1/assignments/*	1. **Forward to** prod-live-bravo-istio-sys-871bc1: 25 (25%) prod-live-charlie-istio-s-212833: 25 (25%) prod-live-delta-istio-sys-185b34: 25 (25%) prod-live-main-istio-syst-deb697: 25 (25%) Group-level stickiness: Off
last **HTTPS 443: default action** *This rule cannot be moved or deleted*	**IF** ✓ Requests otherwise not routed	**THEN** **Forward to** prod-live-bravo-istio-sys-871bc1: **20** (20%) prod-live-charlie-istio-s-212833: **20** (20%) prod-live-delta-istio-sys-185b34: **20** (20%) prod-live-echo-istio-syst-0e304c: **20** (20%) prod-live-main-istio-syst-deb697: **20** (20%) Group-level stickiness: Off

   c. save the rule

Change Destination For static-ip DNS Records

Static-ip DNS records are weighted between the static-ip ALB and the monolith ALB, by default they send all traffic to the static-ip ALB. In case the static-ip ALB is broken, we can redirect all traffic to the monolith ALB.

In case of emergencies:

- Go to https://us-east-1.console.aws.amazon.com/route53/v2/hostedzones#ListRecordSets/Z2WZJFVMQ9OHX6
- Search for “static-ip-api”
- Manually set the static-ip alb weight to 0 and monolith alb weight to 100 for the appropriate record

For non emergencies:

- Navigate to the static-ip directory in the correct terraffirm env, example for us-prod-live: https://github.com/Affirm/terraffirm/blob/master/envs/affirmprod/us-east-1/prod-live/static-ip/alb/inputs.yaml
- Change the primary_alias_weight to 0
- Change the secondary_alias.weight to 100
- Apply

Query ALB Logs

For static-ip logs:
The only prod env where static-ip ALB is set up is in the US

US Prod Live:

- Go to Athena https://us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/
- Execute your query using
  SELECT * FROM "aws_logs"."static_ip_prod_live_logs" where day=’2022/11/21’ limit 10;

For monolith logs:
US Prod Live:

- Go to Athena https://us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/
- Execute your query using
  SELECT * FROM "aws_logs"."monolith_alb_logs" where year='2022' and month='11' and day='21' limit 10;

CA Prod Live:

- Go to Athena https://ca-central-1.console.aws.amazon.com/athena/home?region=ca-central-1#/query-editor/
- Execute your query using
  SELECT * FROM "aws_logs"."monolith_alb_logs" where day='2022/11/21' limit 10;

Target Group Controller Alerts

The target group controller maintains target groups that point at the nodes of our kubernetes clusters, used for routing web traffic. The old flow is:
CDN/partner traffic -> “monolith” ALB -> percentage split between target groups -> istio
For BFCM 2022 we split the ALB into:
CDN traffic -> “monolith” ALB -> percentage split between target groups -> istio
static-ip traffic -> “static-ip” ALB -> percentage split between 2nd set of target groups -> istio

The controller runs in the cosmos-system namespace.

It has the following alerts:
SvcTgtControllerTooManyErrors

- This fires if there are too many AWS errors. It’s normal to see some errors, because the AWS APIs have rate limiting. We can silence this in environments that aren’t ready.

SvcTgtControllerHighMemoryUsage

- High memory usage by the controller. Try restarting it.

SvcTgtControllerNotRunning

- Check pod logs for errors.

SvcTgtControllerNodeCountMismatch

- The number of nodes in the kubernetes cluster does not match the number of nodes seen by the controller. This should auto resolve once the reconciler runs again.

SvcTgtControllerBadTGCount

- We expect both the “main” tier and the “static-ip” tier to have 3 target groups (one each for the traffic, health check, metrics ports). This fires if that’s not the case. Check annotations/ports on the public ingressgateway (in istio-system namespace).
  Should see port 13337 (traffic), 31234 (health check), and one other metrics port

SvcTgtControllerTrafficTierMismatch

- This checks that the number of nodes in the main and static-ip tier target groups are equal. The reconciler should take care of this since it fetches the node set once for all target groups.

Svc-target-controller: Low Targets

- Check logs by running `kubectl -n cosmos-system logs -l app=svc-tg-ctrl`
- IF logs say that there's an issue with fargate nodes, it likely has to do with Cilium:
- Ignore fargate from svc-tg-ctrl

    - Apply label by running: kubectl label node <fargate node name> node.kubernetes.io/exclude-from-external-load-balancers=
    - Turn off kube-applier

Canary Routing for web-ux

Web-ux projects route from the www subdomain CDN -> monolith nginx -> S3. There is a piece of logic which sends requests to a master/ or a canary/ subpath in S3 based on whether the request is hitting a monolith canary pod. Web-ux deploys first to the canary/ S3 paths, and expects canary traffic to flow through the monolith and hit this path for deploy validation. Sometimes, monolith web canary traffic is ramped to zero after a monolith deploy. Use this guide to ramp canary up if they need to do validation and can't get any traffic.
Note: you may run into issues running the job if RPC pods are scaled to zero. The job shifts traffic for both RPC and web, but while web canary pods are always around (they lack an annotation needed for ephemeral canary), RPC canary pods can and do get spun down to zero. In this case, the job will fail because we don't allow traffic to be sent to deployments with zero replicas, and you can manually edit the virtual service or run the vsctl command directly and specify web only.

CDN Runbook

As of March 2023, we have two active CDN providers: AWS Cloudfront and Fastly. We are also actively migrating to Cloudflare. All consoles can be accessed via Okta. Once that migration is complete, this runbook should be updated.

Here's the breakdown of what's in AWS Cloudfront vs Fastly as of March 2023

AWS Cloudfront:

- "www" domain "www.affirm.com", "www.affirm.ca", etc
- api-cf.affirm.com (except CA, where it points directly at the load balancer)

Fastly:

- cdn1.affirm.com (except US prod sandbox, all CA envs - those are in cloudfront)
- cdn-assets.affirm.com

Debugging 403s

403s errors with a Cloudfront error page will most likely be WAF related. Check the path that the 403 is occurring and what origin the request is throwing the 403. We have WAFs at both our CDN layer and the ALB layer. CloudFront can't distinguish between an HTTP status code 403 that's returned by your origin and one that's returned by AWS WAF when a request is blocked.

Page the security team (#ask-security).

Relevant AWS documentation to help troubleshoot: [Troubleshoot 403 errors in CloudFront]()

CDN Cache Invalidation

What is cache invalidation? After an object is cached, it normally remains in the cache until it expires or is evicted to make room for new content. You control the expiration time through standard HTTP headers. You might want to remove an object from the cache prior to its normal expiration time. The usual case we have to do this is when a bad AFJS deploy goes out, because AFJS has a cache expiration of 30 minutes.

AWS Cloudfront

The easiest way to invalidate Cloudfront cache is via the AWS console.

1. Open AWS Console > Cloudfront > Distributions
2. Find the Distribution that you want to invalidate the cache for. Our most "popular" distribution is www (ID: E1S8H7WWZHAQQQ) but this will depend on the URL and might be a different subdomain
3. Go to "Invalidations" on the top navigation bar
4. Click the "Create invalidation" button
5. Enter the path you want to invalidate, try to narrow this path down to as limited scope as you can
6. Click "Create invalidation" and wait for it to complete!

Fastly

The easiest way to invalidate Fastly cache is via the console.

1. Open Fastly via Okta (if you don't have Fastly, cut an IT Support ticket and ping Wilton Wu)
2. For each distribution, there should be a "Purge Cache" button

Querying Cloudfront Logs with AWS Athena

Step 1: Enable Cloudfront Logging to an S3 bucket

- https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/AccessLogs.html
- Can be done in Terraform with "logging_config" https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/cloudfront_distribution
- Go to AWS Cloudfront and then click "Logs" under the "Telemetry" section on the left rail.
- Find and select your distribution by clicking on the corresponding radio button and then click "View details".

- In the “Standard logs” section, if it says “Enabled” under “Status”:
    - On the “Standard logs” section, click the link to the S3 bucket and note the name of the bucket (not the whole URL, e.g. use “affirm-stage-logs” NOT “affirm-stage-logs.s3.amazonaws.com”).
    - Note the S3 bucket prefix if there is any.
- In the “Standard logs” section, if it says “Disabled” under “Status”:
    - On the “Standard logs” section, click “Edit” and choose which bucket you want to save the logs to. If you want to use a new bucket, create the bucket in Amazon S3 first and then come back to this screen. Note the name of the bucket (not the whole URL, e.g. use “affirm-stage-logs” NOT “affirm-stage-logs.s3.amazonaws.com”).
    - (Optional) Type a bucket prefix if you want to use one like “cloudfront/affirm.com/” (the logs will be stored within this subdirectory within the S3 bucket).
    - Toggle the “Status” to “Enabled” and then click “Save changes”.
- Now, your full location for use in Step 2 is “s3://[S3 Bucket Name]/[S3 bucket prefix]/”

Step 2: Create table in AWS Athena to point to logs bucket

- **For BFCM:** Use this PR to copy relevant logs into a new bucket and follow the rest of the doc. Copying logs over will let you run more performant Athena queries.
- https://docs.aws.amazon.com/athena/latest/ug/cloudfront-logs.html
- Go to AWS Athena and click “Explore the query editor”.
- Make sure you have a workgroup selected on the top right.
- In the query editor, copy and paste the below code snippet into the query editor.
- Change the database and/or table name to whatever you want (one that is not currently used). In the example below, “default” is the database name and “cloudfront_logs” is the table name.
- Change the string after “LOCATION” to the full S3 location from Step 1.

```
CREATE EXTERNAL TABLE IF NOT EXISTS default.cloudfront_logs (
  `date` DATE,
  time STRING,
  location STRING,
  bytes BIGINT,
  request_ip STRING,
  method STRING,
  host STRING,
  uri STRING,
  status INT,
  referrer STRING,
  user_agent STRING,
  query_string STRING,
  cookie STRING,
  result_type STRING,
  request_id STRING,
  host_header STRING,
  request_protocol STRING,
  request_bytes BIGINT,
  time_taken FLOAT,
  xforwarded_for STRING,
  ssl_protocol STRING,
  ssl_cipher STRING,
  response_result_type STRING,
  http_version STRING,
  fle_status STRING,
  fle_encrypted_fields INT,
  c_port INT,
  time_to_first_byte FLOAT,
  x_edge_detailed_result_type STRING,
  sc_content_type STRING,
  sc_content_len BIGINT,
  sc_range_start BIGINT,
  sc_range_end BIGINT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LOCATION 's3://CloudFront_bucket_name/'
TBLPROPERTIES ( 'skip.header.line.count'='2' )
```

- If you have issues executing the above command and it says "No output location provided. An output location is required either through the Workgroup result configuration setting or as an API input…"
    - On the left rail, select "Workgroups".
    - Choose your selected workgroup and click "Edit" or create a new one.
    - Under "Query result configuration", choose an S3 bucket to store the query results in. Every query result will be saved to this bucket as a .csv file.
    - Now, go back to the query editor and make sure you are using the workgroup you just edited/created.

Step 3: Query table through AWS Athena

- The fields available for use in queries are seen in the "CREATE TABLE" command that was executed in Step 2. Each field corresponds to a Cloudfront log field specified here: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/AccessLogs.html#LogFileFormat and is in order (i.e. first field on the "CREATE TABLE" command corresponds to the first field under the "Standard log file fields" section in the above link.
- AWS Athena query syntax documentation is here but it is pretty standard SQL syntax: https://docs.aws.amazon.com/athena/latest/ug/ddl-sql-reference.html

- Example queries below

```
-- Get total requests per day to /api/promos/v2/* starting from 08/05/2022
SELECT "date", COUNT(*) total_requests FROM "default"."cloudfront_www_logs"
WHERE uri LIKE '/api/promos/v2/%' AND "date" >= DATE('2022-08-05') GROUP BY
"date" ORDER BY "date";
```

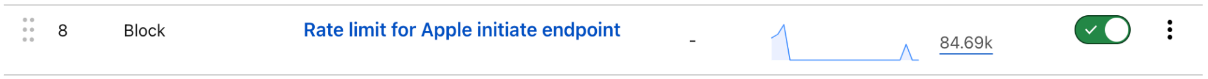
```
-- Get cache hit/miss/error rate for /api/promos/v2/% per day with
percentage
WITH response_results AS (
    SELECT "date", response_result_type, COUNT(*) count FROM
"default"."cloudfront_www_logs" WHERE uri LIKE '/api/promos/v2/%' AND
"date" >= DATE('2022-08-05') GROUP BY "date", response_result_type
), totals_per_day AS (
    SELECT "date", SUM(count) daily_total FROM response_results GROUP BY
"date"
)
SELECT response_results."date", response_result_type, count,
ROUND(count*100.0/daily_total, 2) percent FROM response_results,
totals_per_day WHERE response_results."date" = totals_per_day."date" ORDER
BY "date", response_result_type;
```

Cloudflare Rate Limiting

We have rate limiting set up for our integration with Apple. The rules can be viewed in the Cloudflare console here:

- Prod live: https://dash.cloudflare.com/ce4d5664815cbd558429d45bfaac44d2/affirm.com/security/waf/rate-limiting-rules
- Prod sandbox: https://dash.cloudflare.com/ce4d5664815cbd558429d45bfaac44d2/sandbox.affirm.com/security/waf/rate-limiting-rules

For each rate limit rule in the list, you can see recent activations (click the number for a detailed event view). Each rule has a toggle that can be turned off to allow all traffic through.



8	Block	Rate limit for Apple initiate endpoint	-	84.69k

To troubleshoot, first, make sure the increased traffic triggering the alert is unexpected (no ongoing load tests, etc.). Assuming the backend can handle increased load, we can raise the limit in the console or even disable the rule temporarily. Make sure to revert changes later.

Gringotts Runbook

Decrypting Secrets

To decrypt secrets for a user, first make sure they file a PHELPS ticket for auditing/tracking and to specify which secrets, envs, etc. If it's unclear which package the secret lives in, you can grep for the secret name in the codebase.

You can use either the REPL mode or put all the flags in a single command line to decrypt. Command line form:

```
affirm.gringotts decrypt-secret --repo all-the-things -r <aws region> -a
<aws account> -m <mode> -n <package name> -s <secret name>
```

It may be easier to use the REPL (`affirm.gringotts repl`) as it lists out all options for many of these flags.

Once you have the value, create a new OneLogin secure note (OneLogin portal -> profile picture in top right -> Secure Notes), add the secrets and share it with the person requesting the secrets. Once they confirm they are done checking it, delete the note.

Copying Secrets

If only a specific set of secrets need to be copied, or if they want to change the name of the secret in the destination, use the copy-secret command. Note: this only works when copying a secret between packages in the same environment (AWS account and mode are the same in source and destination). You can use the REPL mode, or the command form like below:

```
affirm.gringotts copy-secret --from-repo <repo> --to-repo <repo> -f
<from_package> -t <to_package> -a <aws account> -r <aws region> -m <mode>
-s <secret name> -n <optional new secret name>  # defaults to same secret
name if -n is omitted
```

If a whole package's secrets needs to be copied to another for every environment you can use affirm.gringotts migrate-secrets

```
affirm.gringotts migrate-secrets -f doubloon/core -t doubloon2/core
```

If there's a unique case (copying to a different country, etc.), then you can run a decrypt-secret command to get the value, save it to a temp file (remember to remove it!) or copy to clipboard, then run an add-secret command to add to the destination env.

Once you're done, put up a PR. You can hand it off to the requesting team to add any configs they need to refer to the secret, etc., and they can merge + deploy. We should also just be able to merge it directly and let them do config changes in a follow up if they want.

Developer FAQs

Send devs to this guide first before answering questions/having them file a phelps ticket

Decrypt Error

```
affirm.platform.gringotts.exceptions.GringottsDecryptError: GringottsRenderer:
Error while decrypting secret: ss://gringotts/cloudwatch_access_key for package
affirm.toolbox
```

Since we attach IamRoleGringottsDecrypt permission to all Iamroles by default as part of K8s components and genesis, this issue should not happen frequently. If it does, most likely AWS_ARN role did not get attached to the pods.

1. If AWS_ROLE_ARN is not present, retry the job.
2. If the role is present, Check if IAM role for this pod has the correct gringotts permissions via AWS console. The Iamrole should have `IamRoleGringottsDecrypt` permissions which should have decrypt access to the gringotts KMS keys.

If that does not work, talk to the COSMOS team but this is not related to Gringotts.

Run the describe pods command and check environment variables:

```
>>> kubectl describe pods <pod name> -n <namespace>

 Environment:
      DEPLOY_CONTEXT_OVERRIDES_vault_auth_role:  investigations
      K8S_NODE:                                   (v1:spec.nodeName)
      KUBERNETES_NODE_NAME:                       (v1:spec.nodeName)
      KUBERNETES_POD_NAME:                       investigations-public-7fdc4f7f79-j4dt9
(v1:metadata.name)
      KUBERNETES_NAMESPACE:                      investigations (v1:metadata.namespace)
      KUBERNETES_POD_UID:                         (v1:metadata.uid)
      AFFIRM_WORKER_ID:                          investigations-public-7fdc4f7f79-j4dt9
(v1:metadata.name)
      DEPLOY_CONTEXT_OVERRIDES_fqdn:             investigations-public-7fdc4f7f79-j4dt9
(v1:metadata.name)
      DEPLOY_CONTEXT_OVERRIDES_app_type:         web
      DEPLOY_CONTEXT_OVERRIDES_is_kubernetes:    true
      DEPLOY_CONTEXT_OVERRIDES_release:          master-117446ef85f40dd7_py3
      PROMETHEUS_MULTIPROC_DIR:                  /tmp
      AWS_STS_REGIONAL_ENDPOINTS:                regional
      AWS_DEFAULT_REGION:                        us-east-1
      AWS_REGION:                                us-east-1
      AWS_ROLE_ARN:
arn:aws:iam::448435121187:role/cosmos/us-east-1-stage-live-bravo/investigations/cosmos_us-east-1-
stage-live-bravo_investigations_investig_d212fe
      AWS_WEB_IDENTITY_TOKEN_FILE:
/var/run/secrets/eks.amazonaws.com/serviceaccount/token
```

DecryptError for salt secrets

```
"GringottsRenderer: Error while decrypting secret:
ss://gringotts/monolith/db/charlotte for package None"}}}]
```

If you notice a package is marked as None in the error, it means it's a secret in salt. For the above case, since the secret is for database, ask someone from platform to run the following command:

```
affirm.gringotts create-key-grant -r <region> -m <mode> -a <AWS account
name> -n gringotts-database-ctrl --repo salt -i <IAM role>
```

Renderer Unavailable

Users have seen this error when setting up a new coordinator or environment. This error is tricky and can be due to many reasons:

```
LoaderError: The renderer jinja | yaml | gpg | gringotts is unavailable, this error
is often because the needed software is unavailable
```

Potential Solution 1

This can happen if the renderer is not in the minion cache. Copy all the custom renderers over:

```
# On the coordinator:
for FILE in `ls /srv/salt/_renderers/`
do
    salt-cp $HOST /srv/salt/_renderers/$FILE
/var/cache/salt/minion/extmods/renderers/
done
```

Potential Solution 2

Check cryptography version on the coordinator and/or minions:

```
$ pip show cryptography
# if cryptography version is NOT 2.6.1, Run
$ pip install cryptography==2.6.1 --upgrade
# sync to necessary minions
$ sudo salt '*' saltutil.sync_all
```

Potential Solution 3

Salt minions don't have gringotts available or the packages it needs (boto3, cryptography==2.6.1) isn’t in the python it uses

```
# on the minions
# run this command to see if there's any useful output about packages being missing
salt-call -l debug state.apply
1. gringotts.py needs to be at /var/cache/salt/minion/extmods/renderers/
       a. find /var/cache -name gringotts.py
       b. can salt-cp it if it doesn't exist
2. cryptography==2.6.1
       a. pip install cryptography==2.6.1
3. boto3 is available (pick version from the working coordinator)
       a. boto3==1.11.17
4. may need to install pyopenssl as well
       a. pip install pyOpenSSL==17.1.0
5. restarting salt-minions sometimes helps
       a. service salt-minion restart
```

note: can go onto the minion and start up python to try to import the packages gringotts need to try to debug.

Potential Solution 4

If the minions can't run any state and the pillar refresh on the master/minions fail, try getting a fresh cache on the master

```
# On the master
cp /var/cache/salt/master/files/base/_renderers/gringotts.py /tmp
systemctl stop salt-master salt-minion
sleep 5

mv /var/cache/salt /tmp/salt.backup
```

```
mkdir -p /var/cache/salt/master/files/base/_renderers/
mkdir -p /var/cache/salt/minion/extmods/renderers/
mkdir -p /var/cache/salt/minion/files/base/_renderers/
cp /tmp/gringotts.py
/var/cache/salt/master/files/base/_renderers/gringotts.py
cp /tmp/gringotts.py /var/cache/salt/minion/extmods/renderers/gringotts.py
cp /tmp/gringotts.py
/var/cache/salt/minion/files/base/_renderers/gringotts.py
systemctl start salt-master
systemctl start salt-minion
```

Expired Affirm Root keys

What do you do if Affirm root keys are about to expire or have expired?
Please follow instructions in Affirm Root keys Rotation Rubric

Deleting Gringotts mutating webhook configuration

If people are seeing errors related to a failed call to the webhook, this will block deploys. To unblock deploys, delete the webhook configuration:

```
# disable kube-applier
kubectl -n kube-system scale deployment/kube-applier --replicas=0
# delete the webhook configuration
kubectl delete mutatingwebhookconfigurations webhook.gringotts-operator.svc.cluster.local
```

Example of webhook errors:

```
error: failed to patch: Internal error occurred: failed calling webhook
"mutate.webhook.gringotts-operator.svc.cluster.local": failed to call webhook: Post
"https://webhook.gringotts-operator.svc:443/mutate?timeout=30s": x509: certificate has
expired or is not yet valid

Internal error occurred: failed calling webhook
"mutate.webhook.gringotts-operator.svc.cluster.local": failed to call webhook: Post
"https://webhook.gringotts-operator.svc:443/mutate?timeout=30s": no endpoints available for
service "webhook"
```

Increase external secrets quota for a specific DT

The default number of external secrets a DT can have is 100, which is defined here:
https://github.com/Affirm/cosmops/blob/0c8ca769f1835dd30b831005219fd1f1cd990cc7/kubernetes/helm/cosmos-system/values.yaml#L18
If a specific DT needs more than that, you need to add an entry to this file and add an override:
https://github.com/Affirm/cosmops/blob/0c8ca769f1835dd30b831005219fd1f1cd990cc7/kubernetes/helm/cosmos-system/templates/dt-ctrl/40-ctrl.yaml#L129

Once you edit that, run `make build_all` from the `kubernetes` directory.

Example PR:
https://github.com/Affirm/cosmops/pull/2695/files#diff-f2be7b9156e1152fb9d7120f0ffdd944cc0335b9eb6c8742bc33cbbdcad2777f

Caching Runbook

Note: this refers to application caches, not celery result stores. Celery result stores are in `terraffirm/envs/<AWS account>/<region>/<environment>/celery/result_store/`.

Dashboards

Cache Cluster Metrics - this gives a view on cluster metrics that we get directly from AWS (server side metrics)
Cache Metrics - dogpile metrics that we get from our cache backends in ATT (client side metrics)

Background

The cache architecture consists of:

- cache clusters (in Elasticache)
    - module defined in tf-caching-module and applied in `terraffirm/envs/<AWS account>/<region>/<environment>/cache/<cache name>/` (example for US prod live)
- DNS records (in Route53)
    - module defined in `terraffirm/modules/cache/dns/` and applied in `terraffirm/envs/<AWS account>/<region>/<environment>/cache/dns/` (example for US prod live)
- client configs (in all-the-things)
    - in `all-the-things/caching/etc/configs/` for Python

- in individual `all-the-things/deployable/<DT>/kubernetes/resources/src/config/deploy_context.yaml` for Kotlin (example for treasury2)

General Metrics and Errors

Overall connection counts across all clusters can be found on Cloudwatch. Cloudwatch metrics for individual clusters and shards can be found in the Elasticache console.
Our biggest weakness is high connection counts. Most clusters have been upgraded to support 100,000 connections, but prod-crypto-redis3 has a limit of 65,000.

Sometimes server-side get/set latencies spike, which may cause a page. In the past, this has resolved itself without any action needed. We could look at turning off these specific alerts, as they don't seem to correlate to client-visible issues.

Rollbar errors related to caching are of the form "`Cache [get/set] error: [reason].`" Details about the deployment, endpoint, etc. are in the metadata. A common error is a node on the server failing over to a replica. These should largely fix themselves, but **if errors persist, rollout restart deployments.** A log of events, including failovers, can be found in the console.

Restarting Deployments

Restarting k8s deployments will often resolve issues with a particular cache. To find which deployments are associated with a particular cache config, check Kibana by querying for:
`payload.evid:dogpile.cache.* and attributes.resource:*<service name>*`
Then, for each deployment, run the following command:

```
kudo kubectl multicluster -f us-east-1-prod-live rollout restart
deployment/<deployment name> -n <namespace>
```

Connecting to Nodes

To connect to a cluster in US prod live, SSH onto `worker-1-3.prod.affirm.com`, which has the Redis CLI installed. Then run
`redis-cli -c -h <primary cluster endpoint>` for the overall cluster
or
`redis-cli -h <node endpoint>` for an individual node.

You can then run CLI commands to debug, run a monitor, etc.

Monitoring Connections

Elasticache has a hard limit of 65,000 connections, which we have raised to 100,000 on most clusters. Alarms will sound and escalate to platform on-call if a cluster hits 85,000 connections.

Adding shards won't help address high connection counts, since the client must connect to all shards. Rollout restart of deployments pointing to a given cluster will temporarily lower connections.

Cache has too many connections

If a cache has too many connections find a deployment that calls into it and restart it with `kubectl -n DT rollout restart deployment <deployment>.`

Creating a new cache

Get the following information from the requesting team:

- A list of all the countries/main envs/modes they need caching in
- **(python only)** The Python service name
- Do they want to store PII in the cache? Discourage it and recommend they use crypto instead. If they really need to store PII in the cache, reuse the crypto cache cluster as the `<cache name>` and skip to step 5 below.

Make sure you have the Terraffirm repository set up locally, you have OneLoginAdmin access in all the relevant AWS accounts, and you're comfortable running Terragrunt commands.

For each country/main env/mode the requesting team needs caching available in, i.e. us-prod-live, do the following:

1. **(us-prod-live only)** Decide if the new cache needs an entire new cluster. For the most part, we can reuse existing clusters but if a particular cluster is running high on connections we shouldn't overcrowd it. If you don't need a new cluster, skip to step 5.
2. Add a new directory in `terraffirm/envs/<AWS account>/<region>/<environment>/cache/` for the new cluster, and add a `terragrunt.hcl` file with the following structure:

terraffirm/envs/<AWS account>/<region>/<environment>/cache/<cache name>/terragrunt.hcl

```
terraform {
  source = "git@github.com:Affirm/tf-caching-module.git?ref=v1.1.11" # check for the most recent release version and update if necessary
}

include {
  path = find_in_parent_folders()
}

locals {
  common_vars = yamldecode(file(find_in_parent_folders("common_vars.yaml")))
}

dependency "monitoring" {
  config_path = "../../../regional/monitoring/onlineservices"
}

dependency "parameter_group" {
  config_path =
"../../../regional/cache_parameter_group/redis-6-cluster-on-high-connections
"
  # this is a special param group giving a 100k connection cap, only
available in us-prod-live
  # for other environments, consult existing cache cluster configs for the
correct param group to use there
}

inputs = merge(
  local.common_vars,
  {
    cache_cluster_id                = "<cluster name>" # these usually
follow the format <environment>-<cache name>-6-v1, e.g.
prod-live-general-6-v1
    cache_cluster_description       = "<description>"
    engine_version                  = "6.x"
    shards                          = 4 # set something reasonable here,
probably don't need more than 4
    replicas                        = 2
    cache_multi_az_enabled          = true
    cache_auto_minor_version_upgrade = true
    cache_param_group               =
dependency.parameter_group.outputs.name
    sns_notification_arn            =
dependency.monitoring.outputs.sns_topic_arn

    enable_security_group = true

    enable_hit_rate_alarm                 = false # turn on once cache is
serving traffic and has good data points
    enable_insufficient_data_alarms       = false # turn on once cache is
serving traffic
    low_cache_hit_rate_alarm_threshold   = 85 # tune for particular cache
    high_set_cmd_latency_alarm_threshold = 60

    max_connections_alarm_threshold = 55000 # use 85000 if environment is
us-prod-live
  }
)
```

3. From this directory, run `terragrunt plan` to make sure everything looks good, then `terragrunt apply` to create the cluster, various CloudWatch alarms, and a security group.
4. Add a dependency block in `terraffirm/envs/<AWS account>/<region>/<environment>/cache/dns/terragrunt.hcl` referencing the new cluster.

   terraffirm/envs/<AWS account>/<region>/<environment>/cache/dns/terragrunt.hcl

   ```
   dependency "<cache name>" {
     config_path = "../<cache dir name>"
   }
   ```
5. Add an entry for the new DNS record in the inputs for `terraffirm/envs/<AWS account>/<region>/<environment>/cache/dns/terragrunt.hcl`, pointing at the proper existing or new cache. The format should be:

   terraffirm/envs/<AWS account>/<region>/<environment>/cache/dns/terragrunt.hcl

   ```
   <service name>-1 = dependency.<cache
   name>.outputs.configuration_endpoint_address
   ```
   where <service name> is the name the team is using in their client configs.
6. From the directory containing `terragrunt.hcl`, run `terragrunt plan` to make sure everything looks good, then `terragrunt apply`. This will create the DNS record in Route53 pointing to the cache cluster, prefixing it with `redis-cache-` and adding it to the `.cache.<internal_zone>` subdomain. Note: for US prod live, we have so many cache clusters that you may see a “too many open files” error. You can increase your file descriptor limit with `ulimit -n 10240`.
7. Set up client configs to point to the DNS record. The DT owner can do this step, following the instructions in the caching user guide. Check their diff to make sure the name matches the name in the DNS record. If the cluster has in-transit encryption turned on (crypto cluster is the main one with this), then make sure SSL is turned on in the client configs.

Splitting Clusters

If a cluster has too many connections, we can split it by changing a cache's DNS record to point to another cluster and restarting deployments. Note that this causes a cold cache and **shouldn't be done for critical caches like orchestration or UCP**.

To split a cluster, find the corresponding DNS module in `terraffirm/envs/{name}/{region}/{environment}/cache/dns/terragrunt.hcl`, modify the record to point to a different cluster (ideally the backup cluster, see below), then run `terragrunt apply`. If time is of the essence, you can manually update the record here. Then rollout restart any affected deployments.

Backup Cluster

We have two backup clusters available. Their DNS addresses are:

- `redis-cache-backup-1.cache.prod.aff`
- `redis-cache-backup-2.cache.prod.aff`

If connection counts become dangerously high on a shared cluster, reroute a service by updating its DNS value in https://github.com/Affirm/terraffirm/blob/master/envs/affirmprod/us-east-1/prod-live/cache/dns/terragrunt.hcl, then run `terragrunt apply`. If time is of the essence, you can manually update the record here. Then rollout restart any affected deployments.

There is also a backup celery result store at `prod-live-celery-results-backup-1.prod.aff`.
Changes to the celery result store require a config change and a redeploy for the service.

Failovers

A node may fail over or be replaced due to maintenance. In this case, a replica will be promoted to be the new primary. Clients will start to time out as they still have connections to the bad node. They may fix themselves over time or may require a restart.

Non K8s Use Cases

The main legacy use case is crypto. It does not usually experience connection pressure but may need to be restarted and unsealed if connection issues occur.

TimeoutError

Possible causes:

- This can happen when a Redis cache failover happens.

- Confirm by looking at the Events tab of ElastiCache of the cache that is timing out; find the name of the cache in the rollbar.

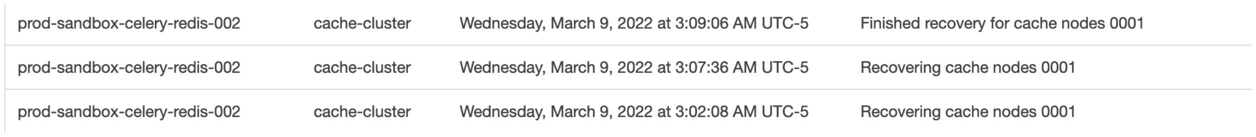
Remediations:

- Ask the owner of the DT which is having this issue to restart the pod
    - Can try this even if there isn't a Redis cache failover

Example rollbar: https://rollbar.com/AffirmMaster/affirm-prod-us/items/603651/

```
#603651 TimeoutError: The operation timed out. Happens when get result from
result_backend: redis://celery-redis.prod-sandbox.aff:6379/ for task:
7471e78f-d2c9-4c97-b78c-f28967e3f1e0
```

Corresponding event in AWS:



prod-sandbox-celery-redis-002	cache-cluster	Wednesday, March 9, 2022 at 3:09:06 AM UTC-5	Finished recovery for cache nodes 0001
prod-sandbox-celery-redis-002	cache-cluster	Wednesday, March 9, 2022 at 3:07:36 AM UTC-5	Recovering cache nodes 0001
prod-sandbox-celery-redis-002	cache-cluster	Wednesday, March 9, 2022 at 3:02:08 AM UTC-5	Recovering cache nodes 0001

Auth Runbook

Dashboards

Auth rpc1 [dashboard]
Auth rpc2 [dashboard] (we only care about 1 endpoint right now)

Database connection issues

We may occasionally see errors like `Max connect failure while reaching hostgroup 2060`. This can happen when the database is overloaded.

Gather metrics:

- Check rollbars on which table is having problems [example].
- Once you know the table, find out which cluster host that table is mapped to using these YAML files [code].
- Check RDS performance insights to see what's going on in the cluster host
- [For rpc2 interfaces] Also check APM traces, for example for user_http_login_v2 to see if something weird is going on.
- Other useful dashboards (URL being sandbox is a misnomer):
    - https://grafana.prod-sandbox-bravo.affirm-keyhole.com/d/J2M2COp7k/proxysql-stats_connection_pool

- https://grafana.prod-sandbox-bravo.affirm-keyhole.com/d/c7qIRppnk/slow-query-finder

Look for the login-sessions table

At this point you'll know which query is causing problems, how widespread it is, and which DT is causing problems. Engage with the DT owner and OnStor oncall to relieve pressure on the database. This can be rate limiting the DT, or increasing proxy-sql instances.

RPC2 Latency High

Check APM traces to see what's causing the latency spike, for example, user_http_login_v2.
If it's the database, check if we're having proxy-sql issues/slow queries (URL being sandbox is a misnomer):

- https://grafana.prod-sandbox-bravo.affirm-keyhole.com/d/J2M2COp7k/proxysql-stats_connection_pool
- https://grafana.prod-sandbox-bravo.affirm-keyhole.com/d/c7qIRppnk/slow-query-finder

Look for the login-sessions table

Follow the above section, OnServ Runbook for troubleshooting DB issues.

If it is not a DB problem, check how long the e2e delays are taking, dashboard. If we're seeing a big difference between client and server timings, it could indicate high gevent contention. Try increasing the number of instances for the rpc2 server. (Note, we're currently serving out of the monolith)

RPC2 Request Count Too High

Based on current users of the user_http_login_v2 rpc2 endpoint, we shouldn't expect enough requests to trigger this alert, if it does that means there's likely a bug in our callers, or there is a very large traffic spike. Check with our callers [dashbaord, check based on aff.rpc.client.* evids] to see what the case is.

Creating Admin API Keys

Some teams want admin API keys to be able to run their services. OnServ can create these keys by:

1. Exec into a monolith web instance for the environment you want to create keys in
2. Open an ipython instance

```
/affirm/bin/ipython
```

3. Run the following setup commands:

```
from affirm.agents.core import Agent
from affirm.models.agents.defs import AppType
from affirm.mdt.entry_points import configure_entry_point
from affirm.mdt.packages import configure_all
from affirm.toolbox2.datastructures.ari.base import ARI
from affirm.toolbox.base62 import random_base62
from affirm.aliases.core.clients.api_key_pair import client as aliases_client
from affirm.schematics.traintrack.defs import Environment
from affirm.toolbox2.datastructures.secured_object import SecuredObject


configure_entry_point('base_app')
configure_all()
```

4. Make a new admin agent and API keys

```
# Fill this out properly!
# example: sally.yen+chameleon@affirm.com
new_agent = Agent.get_or_create_with_email(AppType.admin,
'<your_email>+<name_of_service_which_needs_keys>@affirm.com')

new_agent.verify_email()

public_api_key = ARI.generate()

private_api_key = random_base62(32)

# Use the correct env!
alias = aliases_client.create(AppType.admin, public_api_key,
SecuredObject(private_api_key), Environment.live, new_agent.ari)

aliases_client.verify(alias.ari)

# agent_ari will be in output
# you can print the public and private keys after running above commands
```

5. Share the Agent ARI, public and private keys to the user using 1password vault

Crypto Runbook

Dashboards

Metrics: Chronosphere
Alerts:

- RPC2 errors
- High CPU
- Unhealthy EC2 Hosts
- Kibana Nginx Health Check

High CPU Usage

Configured in Chronosphere UI

Crypto nodes are highly overprovisioned in prod environments usually, so this alert should not trigger unless something is wrong. So it is necessary to investigate why the CPU increased since crypto is in the critical checkout path and any CPU increase can affect our SLAs (especially if it is prod live environments)

1. If this alert is triggered, check the Crypto Service dashboard for the environment the alert is triggered for. Specifically check the Operation counts and if you see an increase in decrypts which is a CPU intensive operation.

   If High CPU is due to an increase in traffic (mostly between 10-2 PM PST), it is probably a valid reason for this alert. Since Crypto nodes are over-provisioned and CPU threshold is set to a low value in the alert, we should still be okay. You can wait for a few minutes/not take any action.

2. If the traffic is not high, and operation count is the same, then check Cloudwatch EC2 metrics on AWS console to make sure the nodes are okay and there are no AWS related system errors or NLB errors.

Note: If this alert is very noisy and is happening pretty frequently and the CPU is usually going above the set threshold, think about increasing the node count in prod environments (message in #ask-infrastructure-engineering).

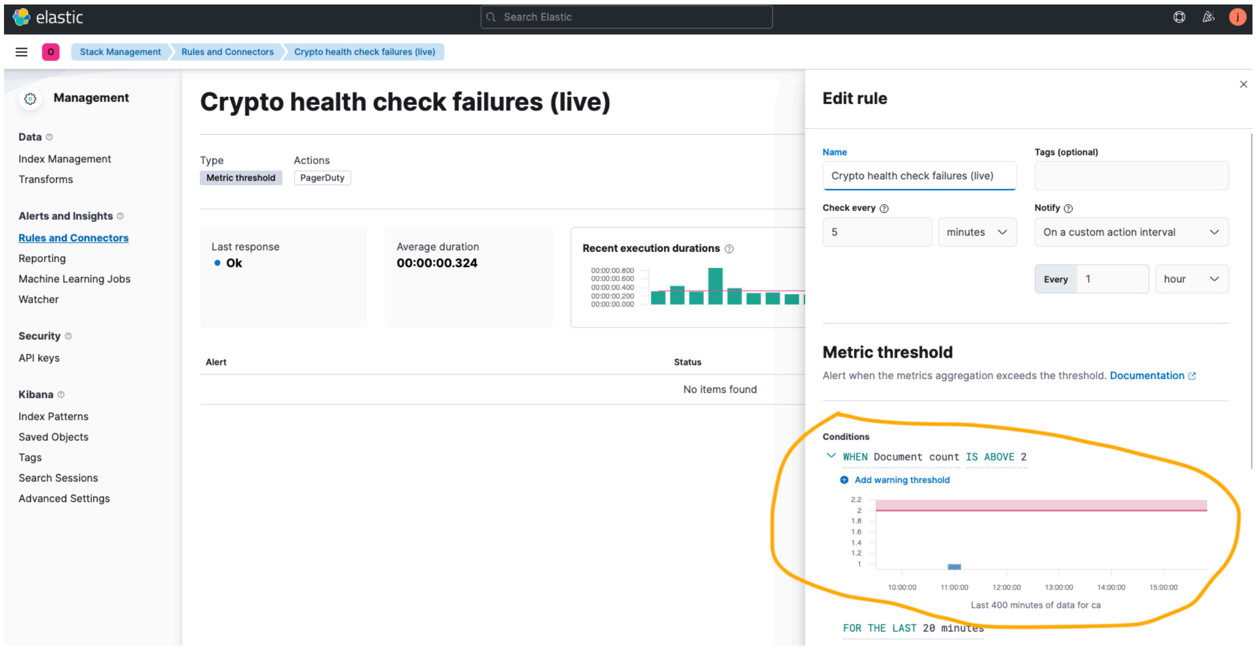
Unhealthy EC2 Hosts

If you get paged for this, please check the target group status here (Make sure to select the correct AWS account/region you got paged for). If the instance you got paged for is showing as healthy, check the unseal status of the node, by SSHing into the node.

Crypto health check failures - Document count is greater than a threshold

This alert is set up in Kibana. It is an nginx health check that can't be set up in Chronosphere. It is a low priority alert that won't be needed after K8s migration and is also pretty flakey. If there are no rollbars for crypto (check #bot-crypto-rollbar-prod in Slack), then there shouldn't be an issue.

This alert may not self-resolve in PD, so you may need to resolve. You can check if the count has decreased by going to the alert in Kibana and clicking edit. You should see a graph:



Crypto health check failures (live)

Type: Metric threshold
Actions: PagerDuty

Last response: Ok
Average duration: 00:00:00.324

No items found

Edit rule

Name: Crypto health check failures (live)
Tags (optional):
Check every: 5 minutes
Notify: On a custom action interval
Every 1 hour

Metric threshold
Alert when the metrics aggregation exceeds the threshold. Documentation

Conditions
```
WHEN Document count IS ABOVE 2
FOR THE LAST 20 minutes
```

This is a flaky alert because if the traffic is high and the inbound socket is full with RPC requests and healthchecks, sometimes the server does not respond back in time to the client and this kibana alert starts firing.

Filter: environment.mode: "live" and host.name: crypto- and url.original: "/_health" and not http.response.status_code: 200*

Things to check if this alert fires:

1. Check rollbar for the environment this alert is firing for. If you see any `key loading errors` or `RPC request failures` or any 5xx from crypto, there is an actual issue.

2. Also check chronosphere dashboard to make sure all requests are fine.

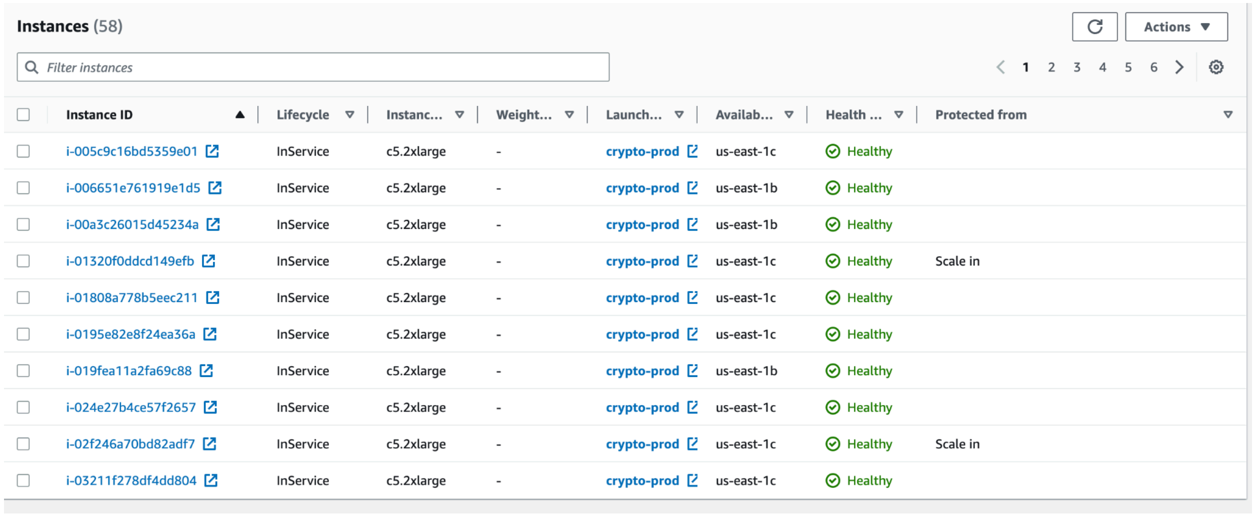
The alert should self resolve after sometime when traffic subsides. If not, check pod logs or EC2 logs to see if there is some other issue.

ASG ScaleUp (Bring up new Crypto EC2 nodes)

Crypto uses AWS Auto-scaling groups (ASG) to scale up nodes which runs a bunch of scripts/commands on the salt coordinator and brings up the nodes.
If the nodes don't get setup (Check teleport to see if you can ssh into the node or if the hostname is setup correctly), then follow steps here.
If you are unsure about this, you can contact InfraOps team to get help on scaling up the nodes.

We should have "Scale In" Protection on for the crypto nodes and if we don't, add it to the instance or ask the infraops team to update that. This prevents new nodes from coming up in case of node cycling because if a new node comes up, it requires a "manual unseal".

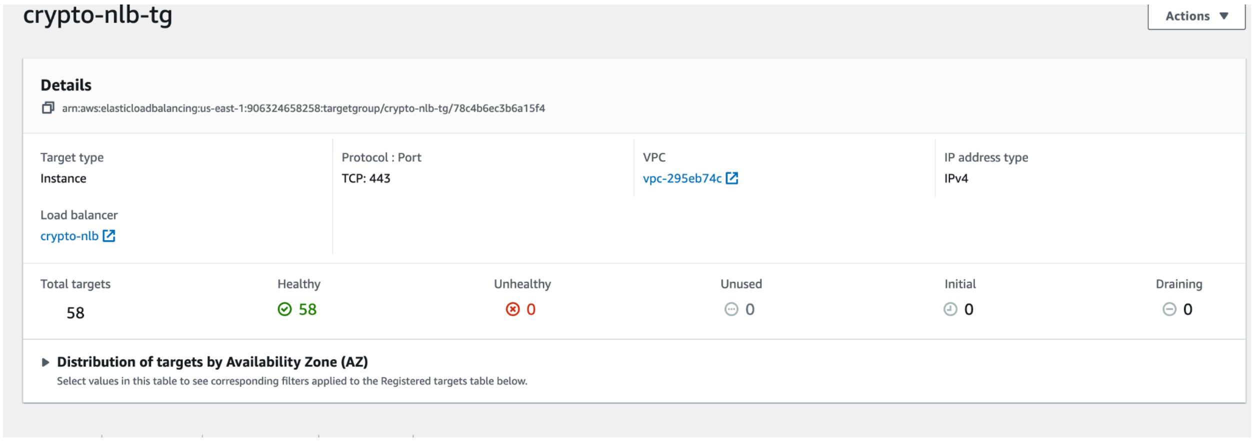
You can check "Activity tab" on ASG page on the AWS console to check the status of a node bootstrap.



Instance ID	Lifecycle	Instanc...	Weight...	Launch...	Availab...	Health ...	Protected from
i-005c9c16bd5359e01	InService	c5.2xlarge	-	crypto-prod	us-east-1c	Healthy	
i-006651e761919e1d5	InService	c5.2xlarge	-	crypto-prod	us-east-1b	Healthy	
i-00a3c26015d45234a	InService	c5.2xlarge	-	crypto-prod	us-east-1b	Healthy	
i-01320f0ddcd149efb	InService	c5.2xlarge	-	crypto-prod	us-east-1c	Healthy	Scale in
i-01808a778b5eec211	InService	c5.2xlarge	-	crypto-prod	us-east-1c	Healthy	
i-0195e82e8f24ea36a	InService	c5.2xlarge	-	crypto-prod	us-east-1c	Healthy	
i-019fea11a2fa69c88	InService	c5.2xlarge	-	crypto-prod	us-east-1b	Healthy	
i-024e27b4ce57f2657	InService	c5.2xlarge	-	crypto-prod	us-east-1c	Healthy	
i-02f246a70bd82adf7	InService	c5.2xlarge	-	crypto-prod	us-east-1c	Healthy	Scale in
i-03211f278df4dd804	InService	c5.2xlarge	-	crypto-prod	us-east-1c	Healthy	

If a new node does come up, check the following to make sure it is working as expected:
1. SSH into the node using teleport and run
    a. sudo -Es; unseal
    b. If everything is fine and the node is unsealed, you will see "Keyring is unsealed, nothing to do"
    c. If the node is sealed, please enter the master password. For Prod live environments, ask Shyam Mani to enter the master password, OR get 2 key custodians to enter their key shares to unseal the nodes.

2. Check the [Target Group](#)status. Make sure the new nodes are registered here on the target group.


crypto-nlb-tg

Details

arn:aws:elasticloadbalancing:us-east-1:906324658258:targetgroup/crypto-nlb-tg/78c4b6ec3b6a15f4

Target type	Protocol : Port	VPC	IP address type
Instance	TCP: 443	vpc-295eb74c	IPv4

Load balancer
crypto-nlb

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
58	58	0	0	0	0

Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.

To manually register a node to a target group, do the following:

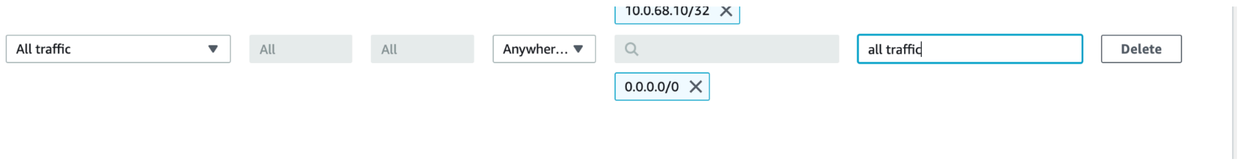
PLEASE MAKE SURE THE NODE IS UNSEALED before adding the node to the Target group.

1. Click on the target group for the correct environment (live or sandbox).
2. Click “Register targets” and add the new instance.
3. It will be in “pending” state and will eventually get registered.

Check the chronosphere dashboard and rollbar to make sure the nodes are healthy.

Outbound Security Group

Egress rules for crypto are pretty restrictive and we disallow all traffic, except certain endpoints. But when we deploy crypto, it needs to fetch images from Stage ECR and we **temporarily** have to add a rule to allow all outbound traffic. This is the rule you should add and MAKE SURE YOU DELETE IT after you are done with deploy.


All traffic | All | All | Anywher... | 10.0.68.10/32, 0.0.0.0/0 | all traffic

After adding this rule, you can run this command on the coordinator to make sure the nodes are able to reach ECR repo and authenticate with it:

```
>> salt 'crypto*' cmd.run '/usr/local/bin/affirm.ecr login  --account-id 448435121187
--assume-role ec2-docker-client --region us-east-1'
```

Getting access to Crypto

You need to be a Key Custodian (have keyshare to unseal crypto) to get this access. Please reach out to Wilton in case you need it. If you are a key custodian and cannot ssh in to the node, do the following:

- Check your salt config here (users.sls)
- Create a ticket for InfraOps to get access to nodes via teleport.

Testing RPC call to crypto from monolith pod

- Get an ARI from one of the crypto DynamoDB tables
- Exec into a monolith RPC pod
- Save script below in a file and then run it:

```
# Run this with /affirm/bin/python from within the monolith virtualenv
(docker-shell-frontend)
# Input is newline-separated crypt ARIs, output is a CSV with fields [crypt
ARI, decrypted SSN]
# Adjust filenames as necessary
# This is using decrypt_batch. But you can change that you just use
client.decrypt(ari)
from affirm.crypto4.client import client
from affirm.mdt.entry_points import configure_entry_point
from affirm.platform.logger import AffirmLoggerInit
from affirm.crypto4.rpc.xceptions import ARINotFound, InvalidARI

AffirmLoggerInit()
configure_entry_point('base_app')

batches = [[]]
with open('crypt-ari.txt') as f:
    for ari in f:
        if len(batches[-1]) == 100:
            batches.append([])
        batches[-1].append(ari.strip())

with open('decrypted.txt', 'w') as f:
    for i, batch in enumerate(batches):
        decrypted_list = client.decrypt_batch(batch)
        # assert len(decrypted_list) == len(batch)
        for ari in batch:
            result = decrypted_list[ari]
            # if isinstance(result, dict):
                # result = result['ssn']
            f.write('{},{}\n'.format(ari, result))
        print('Decrypted batch {} of {}'.format(i + 1, len(batches)))

def retry_decrypt_batch(ari_list, decrypted_list):
    # type: (List, Dict[text_type, text_type]) -> None
    """
    Retry each ARI individually instead of a batch
    """
    failed = 0
    for ari in ari_list:
        try:
            decrypted_list[ari] = client.decrypt(ari)
        except InvalidARI or ARINotFound:
            # non-retryable errors
            failed += 1
    return
```

Crypto ARI not found error

It’s possible that DT owners might have used the mock crypto client when encrypting, so the crypto ARI would not exist in this case. (This is only true for Stage CA for some/most Kotlin clients) but this is not the case for python. Also in some cases, payments team does redact some ARIs from dynamo as they have some jobs running at their end which can also lead to an error like this.

1. Ask the DT owner if they have the failing ARI, or check rollbar.
2. Check if that ARI is present in DynamoDB crypto table
    a. If it is not present, then this is a valid error.
    b. Else, something else might be wrong and would need further investigation.

Sometimes you might see ciphertext empty but ARI being present in the DynamoDB. This is because that ARI might have been **redacted** by one of the Payments Job and the db row will have a `removed_at` field in that case as well. This is a valid use case and if this happens, you will see a rollbar with **KeyError**: key[‘ciphertext’]

Redact Crypto Hash ARI

In some cases, hashed data may need to be redacted (example ticket). The crypto `redact` API does not work for hashed text (see ticket for more info). Run this script to redact hash text manually:

```
import boto3
from datetime import datetime
from decimal import Decimal
from affirm.crypto4.service.repo.dynamodb import _with_base_schema

session = boto3.session.Session(profile_name=AWS_PROFILE)
dynamodb = session.resource('dynamodb', region_name=AWS_REGION)
table = dynamodb.Table(CRYPTOHASH_TABLE_NAME)

ari = CRYPTO_HASH_ARI

response = table.get_item(
Key=
{ 'ari': ari, }
)
print(response['Item'])

result = table.update_item(
Key={'ari': ari},
AttributeUpdates=_with_base_schema(
{ 'hash_text': \{'Action': 'DELETE'}
,
'removed_at': {'Value': Decimal(datetime.utcnow().timestamp())},
}, for_update=True),
)
print(result)
```

Crypto on Kubernetes

Getting access to Crypto

You need to be a ***CryptoAdmin*** to be able to perform operations on the pods. Please create an IT ticket to get access if you require it.
You will also have to sign a KeyCustodian form if you need this access and to do that, please reach out to the security team (GRC team in #ask-security).

Running exec on Crypto

Whenever a CryptoAdmin performs *exec* operation on the crypto pods, please make sure to tag security team and *#prodops* in the *#prod-issues-eng* channel. This is because we have setup alerts in case there is an exec access on crypto pods and if that happens, an alert will be triggered and Security team will be notified.

Create nodegroups for Crypto

Crypto nodes need to have restricted access, for instance, restricted egress access for PCI and SOC compliance reasons as it stores Cardholder data and we chose to put them in a separate nodegroup so we can control and manage various operations such as when the nodes cycle, which pods get scheduled on the nodes etc.

This section covers commands that are needed to create these dedicated nodegroups. The nodegroups code resides in terraffirm cosmos module here and this is one of the sample PRs. We could not just run terragrunt apply as there is a dependency between creating the iamroles and policies and creating the nodegroups. Terragrunt fails unless the nodegroup is created in this order.

Below commands are for affirm-stage, change it based on your cluster and AWS account you need to run the command for:

```
AWS_PROFILE=affirm-stage terragrunt apply
-target=module.bravo_cluster.aws_iam_role.k8s_worker

AWS_PROFILE=affirm-stage terragrunt apply
-target=module.bravo_cluster.aws_iam_role_policy_attachment.AmazonEKSWorker
NodePolicy

AWS_PROFILE=affirm-stage terragrunt apply
-target=module.bravo_cluster.aws_iam_role_policy_attachment.AmazonEC2Contai
nerRegistryReadOnly
-target=module.bravo_cluster.aws_iam_role_policy_attachment.AmazonEBSCSIDri
verPolicy
-target=module.bravo_cluster.aws_iam_role_policy_attachment.AmazonEKS_CNI_P
olicy

AWS_PROFILE=affirm-stage terragrunt apply
-target=module.bravo_cluster.aws_eks_node_group.managed_nodegroups
```

After running these commands, apply the rest of changes using Spacelift, as there are more resources that need to be created such as alarms, propagating tags, and ASGs.

Crypto Pod Count High

This alert was added because of a hard connection limitation on SQL side.
Crypto uses SQL db for the Password table which is mainly used by merchants to create/check/update passwords. Unlike other DTs, crypto does not use ProxySQL which can do better connection management and rather uses “sql_store” library to make direct connection to RDS instances.
Due to this reason, the connection management/pooling capabilities are limited and lead to a very high connection count on SQL side. AWS RDS has 16,000 hard connection limitation after which there can be query degradation and other side-effects to the clients.

SQL connections can be monitored here:

https://affirm.chronosphere.io/dashboards/d/online-storage-onstor-database-monit/onstor-aurora-mysql-metrics-per-instance?orgId=1&from=now-7d&to=now&var-cloud_region=us-east-1&var-environment=prod&var-instance=affirm8-us-east-1-primary-1&var-instance=affirm8-us-east-1-primary-2

If this alert gets fired, check the following:

1. Work with **#ask-online-storage** team to monitor the above SQL connections dashboard.
2. If the connection limit is lower than 14-5K, then check if the traffic spike is temporary and if pods will scale down eventually. In that case, no action is needed.
3. If the HPA is not scaling pods down after waiting for a couple minutes (~10-15 mins) and connection count on SQL is increasing as well, then rollout restart crypto pods to bring down the connections.

Please do 3. only if it is necessary and after consulting with the Onstor team.

We have vertically scaled crypto pods by a lot, so this alert should not trigger frequently. Apart from the SQL connections, also check general Crypto statistics like CPU, dynamoDB latency to make sure this scaleup is not due to some other issue in the infrastructure.

Crypto Service K8s - High CPU Usage/ Crypto: Container CPU is high

1. If this alert is triggered, check the Crypto Service dashboard for the environment the alert is triggered for. Specifically check the Operation counts (under RPC2 statistics) and if you see an increase in operation counts (encrypts/decrypts), that would be a valid reason for high cpu and they are CPU intensive operations.
2. If High CPU is due to an increase in traffic (mostly between 10-2 PM PST), it is probably a valid reason for this alert and crypto pods should scale up in this case.
   Check HPA to see if pods are ready to be scaled up:

```
> kubectl get hpa -n crypto
```

3. Check events logs in the cluster by running the command below or check Kibana for kubelet logs

```
>  kubectl get events -n crypto
```

If the alert is not due to the above reasons or you cannot find a reason, there might be something wrong with the node.

- Check `top` by exec-ing into the pod to see which process is taking up CPU.
- Check the node on which this pod is scheduled to see if the node is ok. Details on debugging nodes can be found in COSMOS runbook or contact the COSMOS team for more information (#ask-kubernetes Slack channel).

```
❯ kubectl get pods -n crypto -o wide
NAME                                                     READY   STATUS      RESTARTS   AGE     IP
NODE                                            NOMINATED NODE   READINESS GATES
crypto-rpc-69bc59ddb6-qkz5m                              4/4     Running     0          10h
10.3.100.141   ip-10-3-111-219.ca-central-1.compute.internal   <none>           <none>
crypto-rpc-69bc59ddb6-wbqjg                              4/4     Running     0          10h
10.3.228.245   ip-10-3-238-20.ca-central-1.compute.internal    <none>
```

Crypto: Karpenter Provisioner Usage High

Karpenter is used by Crypto (and other DTs) to manage node resources and pod scheduling. If you get paged for this alert, most likely we are reaching the threshold and if crypto pods autoscale more, it could lead to these resources (CPU/memory) getting exhausted and pods failing to get scheduled.

This happens during crypto deploys for a small period of time. You can check if the increase coincides with a deploy here. If the deploy is failing because pods cannot be scheduled on crypto nodes, increase the CPU/memory on karpenter provisioner like this PR and deploy cosmops change. If the usage goes down once the deploy is finished, then no need to increase the limit.

If there is no crypto deploy going on, check if crypto pods are autoscaling consistently and if there is high traffic on crypto pods. If that is the case, increase the CPU/memory on karpenter provisioner like this PR and deploy cosmops change.

Make sure to let COSMOS team know in #ask-kubernetes that you are making this change and increasing resources, as there could be a quota/threshold on these resources as well and get the PR reviewed by COSMOS team.

For a small spike and pos scaling back down, no need to increase the resource limits.

Crypto: Container Throttling

ContainerThrottling can happen for a variety of reasons. In order to figure out the error, run

```
> kubectl describe pod <pod-name> -n crypto
```

A common problem that may happen is Vault may be “slammed” with high load (bobapki container). If the Last State for bobapki container is Terminated, the Reason is *Error*, and the Message is

```
time="2021-06-21T19:19:59Z" level=fatal msg="certificate issue
failed" addr=vault.prod.aff crt=/nail/var/boba-pki-cert.pem err="aws
login request send failed: Post
\"https://vault.prod.aff/v1/auth/aws/login\": context deadline
exceeded (Client.Timeout exceeded while awaiting headers)"
key=/nail/var/boba-pki-key.pem role=worker self-signed=false
tag=vault_err
```

Check CPU utilization on Vault and see if there was a spike. If there was a spike, we know the problem is with Vault. In this event, contact someone who knows about Vault or BobaPKI setup to help resolve this (#ask-infrastructure-engineering).

This might need further investigation if we see app pods starting to throttle and we might have to increase the CPU requests/limits that we have allocated to the pod.

If the reason for termination is a different error, please update this runbook with how you resolved this.

Crypto: HPA replica is exhausted

This means the max HPA has reached this cluster.

1) Examine the HPA by running the command in the cluster and you can see the reason for recent scaleup. We scale up either on nginx connection count or CPU and most likely for crypto, it is the CPU. Check the Chronosphere dashboard if CPU has been high due to high traffic/operations on the crypto pods

```
> kubectl get hpa <hpa-hittinglimit-name> -n crypto
> kubectl describe hpa <hpa-hittinglimit-name> -n crypto
```

2) If there is a valid reason for Scale-up and you have reached the max, you will have to manually update the max HPA replica to quickly address the issue.
    a) Disable kube-applier in this cluster

```
> kubectl -n kube-system scale deployment/kube-applier --replicas=0
> For nodes that are not managed by Keda:
kubectl -n <ns> edit hpa <hpaHittingLimit>
```

```
> For nodes that are managed by Keda
kubectl multicluster -f us-east-1-prod-live -n crypto patch so
crypto-rpc --type='merge' -p '{"spec":{"minReplicaCount": 4,
"maxReplicaCount": 25}}'
```

   b) Double the maxReplicas (or set it to whatever seems reasonable based on the metrics HPA is showing. For example if you see 50%/10% for CPU on the HPA, 5x might be appropriate.

3) Determine if there any impact from this limit being hit: Are latencies/errors on the service the HPA is managing observable?
4) If it doesn't seem like there is a valid reason for the scaleup, it might be indicative of a code issue (ex: some code deploy or version bump caused significantly higher CPU usage).
5) Commit the new HPA max into code if this is a valid case in all-the-things and deploy.
6) Turn kube-applier back on:

```
> kubectl -n kube-system scale deployment/kube-applier --replicas=2
```

Unauthorized access to Crypto pods

Link to the doc

Cache GET/SET p99 latency

We have not seen this alert being fired ever, so there is not much to add here. But some things to check in case this happens:

- Check Crypto cache metrics on AWS console, specifically for CPU/memory.
    - CPU and memory for Crypto cache is usually very low.
    - Check cache Hit/miss rate. It is possible that there are a lot of cache misses which is adding to latency. This might be an intermittent issue due to cold cache. If it persists, maybe the TTL on the cache is too low and might be worth checking.
- Check if there was some cache failover or maintenance on Elasticache which has downgraded the performance.
- Make sure, if it is a new endpoint, that we are using the cache API correctly.

Crypto RPC request timeouts

1. Look at internal ingressgateway pod logs to make sure there are no errors and the requests look the logs below

```
> kubectl get events -n crypto
This command can show if any events/errors occurred in crypto namespace.
Check to make sure everything looks good.

> kubectl get pods -n istio-system -o wide | grep internal

> kubectl logs -f <internal-ingressgw-pod-name> -n istio-system
160.145:8443 10.0.184.106:25928 crypto.stage.aff -
[2023-09-01T17:30:18.117Z] "- - -" 0 - - - "-" 5452 8132 49580 - "-" "-" "-" "-"
"10.0.210.228:443" outbound|443||crypto-rpc.crypto.svc.cluster.local 10.0.160.145:60924
10.0.160.145:8443 10.0.71.149:6643 crypto.stage.aff -
[2023-09-01T17:27:04.293Z] "- - -" 0 - - - "-" 5452 8132 246942 - "-" "-" "-" "-"
"10.0.210.228:443" outbound|443||crypto-rpc.crypto.svc.cluster.local 10.0.160.145:36570
10.0.160.145:8443 10.0.73.221:13481 crypto.stage.aff -
[2023-09-01T17:26:15.317Z] "- - -" 0 - - - "-" 5406 8167 301693 - "-" "-" "-" "-"
"10.0.248.166:443" outbound|443||crypto-rpc.crypto.svc.cluster.local 10.0.160.145:57684
10.0.160.145:8443 10.0.91.137:64898 crypto.stage.aff -
[2023-09-01T17:26:42.610Z] "- - -" 0 - - - "-" 5466 8129 301400 - "-" "-" "-" "-"
"10.0.248.166:443" outbound|443||crypto-rpc.crypto.svc.cluster.local 10.0.160.145:44142
10.0.160.145:8443 10.0.65.230:32696 crypto.stage.aff -
[2023-09-01T17:25:45.720Z] "- - -" 0 - - - "-" 13883 11454 370944 - "-" "-" "-" "-"
"10.0.210.228:443" outbound|443||crypto-rpc.crypto.svc.cluster.local 10.0.160.145:34834
10.0.160.145:8443 10.0.184.106:39745 crypto.stage.aff -
[2023-09-01T17:28:28.803Z] "- - -" 0 - - - "-" 5313 8167 301207 - "-" "-" "-" "-"
"10.0.248.166:443" outbound|443||crypto-rpc.crypto.svc.cluster.local 10.0.160.145:53150
10.0.160.145:8443 10.0.71.149:46776 crypto.stage.aff -
[2023-09-01T17:29:21.272Z] "- - -" 0 - - - "-" 6243 8409 301738 - "-" "-" "-" "-"
"10.0.210.228:443" outbound|443||crypto-rpc.crypto.svc.cluster.local 10.0.160.145:56172
10.0.160.145:8443 10.0.84.67:9669 crypto.stage.aff -
[2023-09-01T17:26:16.919Z] "- - -" 0 - - - "-" 14869 12102 495332 - "-" "-" "-" "-"
"10.0.210.228:443" outbound|443||crypto-rpc.crypto.svc.cluster.local 10.0.160.145:36296
10.0.160.145:8443 10.0.217.60:7241 crypto.stage.aff -
[2023-09-01T17:26:26.788Z] "- - -" 0 - - - "-" 10201 10102 496221 - "-" "-" "-" "-"
"10.0.248.166:443" outbound|443||crypto-rpc.crypto.svc.cluster.local 10.0.160.145:57290
10.0.160.145:8443 10.0.90.44:32254 crypto.stage.aff -
```

2. If internal-ingressGW has issues, make sure the virtual Service and Gateway resources look correct and correct routing is setup.
   a. For Gateway resource: Make sure it is a TLS Passthrough Mode and Hosts are set to correct information as mentioned below.
   b. For VirtualService: Make sure SNI hosts match r53 record name and destination is set to Crypto service `crypto-rpc.crypto.svc.cluster.local`

```
> kubectl get vs crypto-rpc -n crypto -o yaml
  spec:
    gateways:
    - istio-system/internal-gateway
    hosts:
    - crypto.stage.aff
    - crypto.stage-live-bravo.stage.aff
    tls:
    - match:
      - port: 443
        sniHosts:
        - crypto.stage.aff
        - crypto.stage-live-bravo.stage.aff
      route:
      - destination:
          host: crypto-rpc.crypto.svc.cluster.local
          port:
            number: 443
        weight: 100
      - destination:
          host: crypto-rpc-canary.crypto.svc.cluster.local
          port:
            number: 443
        weight: 0

❯ kubectl get gateway internal-gateway -n istio-system -o yaml
spec:
  selector:
    app: internal-ingressgateway
  servers:
  - hosts:
    - '*.stage.aff'
    - '*.stage-live-bravo.stage.aff'
    port:
      name: tls
      number: 443
      protocol: TLS
    tls:
      mode: PASSTHROUGH
  - hosts:
    - '*.stage.aff'
    - '*.stage-live-bravo.stage.aff'
    port:
      name: mysql
      number: 3306
```

3. Lastly check the Crypto Service endpoints
    a. Make sure endpoints are set correctly to Pod IPs in crypto service. THis will ensure any traffic reaching this service will go to the correct pods.

```
> kubectl describe service crypto-rpc -n crypto
Name:              crypto-rpc
Namespace:         crypto
Labels:            app=crypto-rpc
```

```
                      metrics.affirm.com/scrape-dt-metrics=true
Annotations:          components.cosmos.affirm.com/component: rpc2
                      components.cosmos.affirm.com/language: python
                      components.cosmos.affirm.com/version: v32
Selector:             app=crypto-rpc
Type:                 ClusterIP
IP Family Policy:     SingleStack
IP Families:          IPv4
IP:                   172.20.227.199
IPs:                  172.20.227.199
Port:                 https  443/TCP
TargetPort:           443/TCP
Endpoints:            10.0.210.228:443,10.0.248.166:443   >>>>>>>>>>>>>>>
Port:                 afrm-mtrx-app  6003/TCP
TargetPort:           6003/TCP
Endpoints:            10.0.210.228:6003,10.0.248.166:6003
Port:                 afrm-mtrx-nginx  8080/TCP
TargetPort:           8080/TCP
Endpoints:            10.0.210.228:8080,10.0.248.166:8080
Session Affinity:     None
Events:               <none>

> kubectl logs -f <pod-name> -n crypto -c <container name>
Lastly check if you can see errors in the app container causing application level
errors.
```

Security Groups

Crypto Stuck on PodsInitializing or failing to come up (ContainerRestartsTotal)

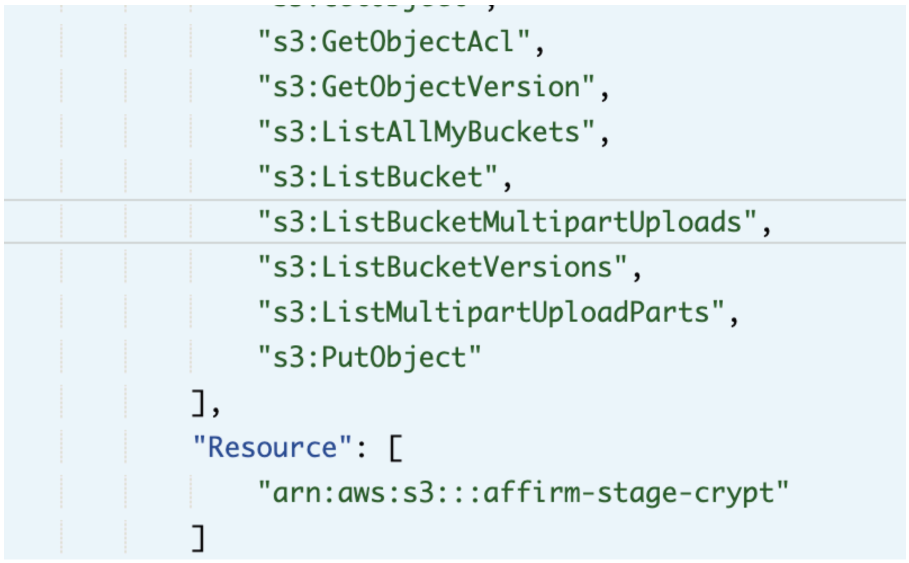
There could be multiple reasons for this. Check if init containers came up by running describe pod or checking logs for those containers.

```
> kubectl describe pod <pod-name> -n crypto
> kubectl logs -f <pod> -n crypto -c <container name>
```

1. Check `load-rds-cert-bundle` container to make sure that it is able to successfully fetch the RDS cert. It runs a curl command to *`truststore.pki.rds.amazonaws.com`*, so check if this container is blocking the pod from coming up.
2. Check `bobapki-1` container to make sure pods can fetch cert from vault.
3. If there is an error in app container due to `DecryptionFailed` or `Error fetching S3 Object`:
    a. Check `app` container to make sure connection to S3 exists:
        i. Install awscli on the pod: *apt-get install awscli*

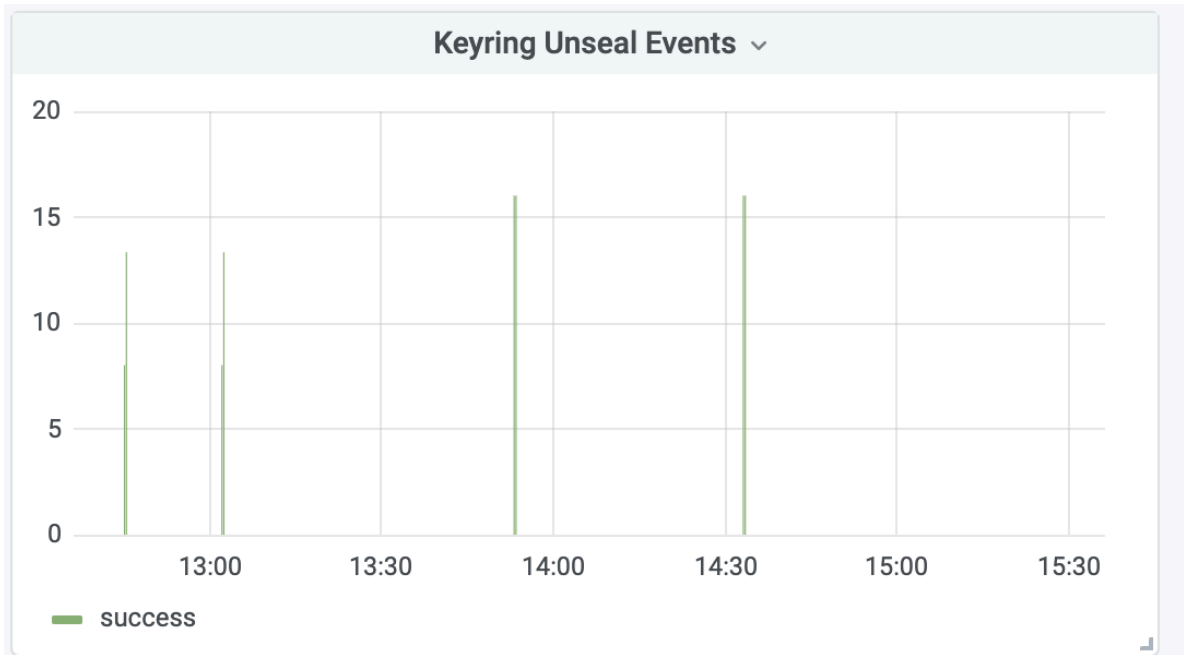
ii. Run: *aws s3 sync s3://affirm-<env-name>-crypt/crypt2/encrypt/CARD.json .*

b. If S3 sync is failing, make sure to check Crypto IAM role on AWS console and verify that it has access to Crypto's S3 bucket (for the appropriate environment).



```
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListBucketVersions",
                "s3:ListMultipartUploadParts",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::affirm-stage-crypt"
            ]
```

4. Maybe pods are **not Unsealed**. Check Chronosphere dashboard for Unseal status. You should see an Unseal event if there was a pod restart. If that does not happen, most like the pod is Sealed and is not ready to accept traffic yet (Health checks to this pod will continue failing)

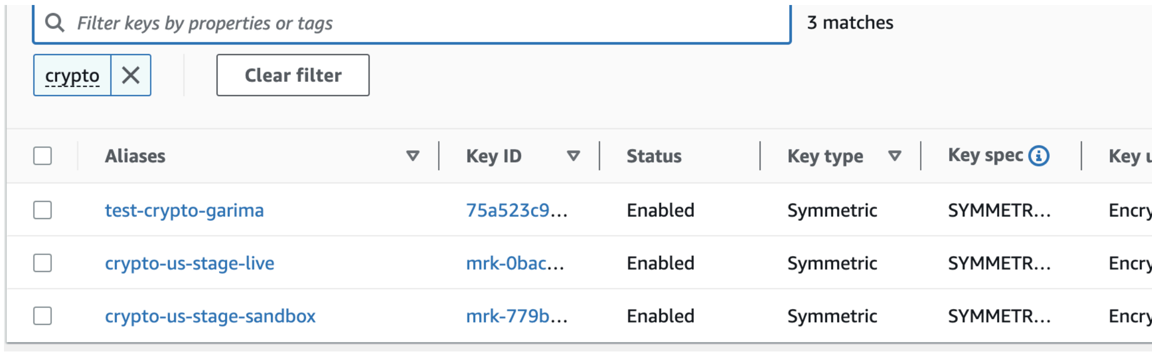


If the pods are sealed, check logs to see if you can find a reason. Maybe there is an issue with AWS KMS Key decryption.

a. If the pods fail to decrypt, check the AWS KMS console to verify that the key policy is accurate:

i. Check the AWS console to verify KMS key status (*Search for crypto in AWS KMS to list crypto keys used.* **Note**: You need to be a **CryptoAdmin** to view this key!!).

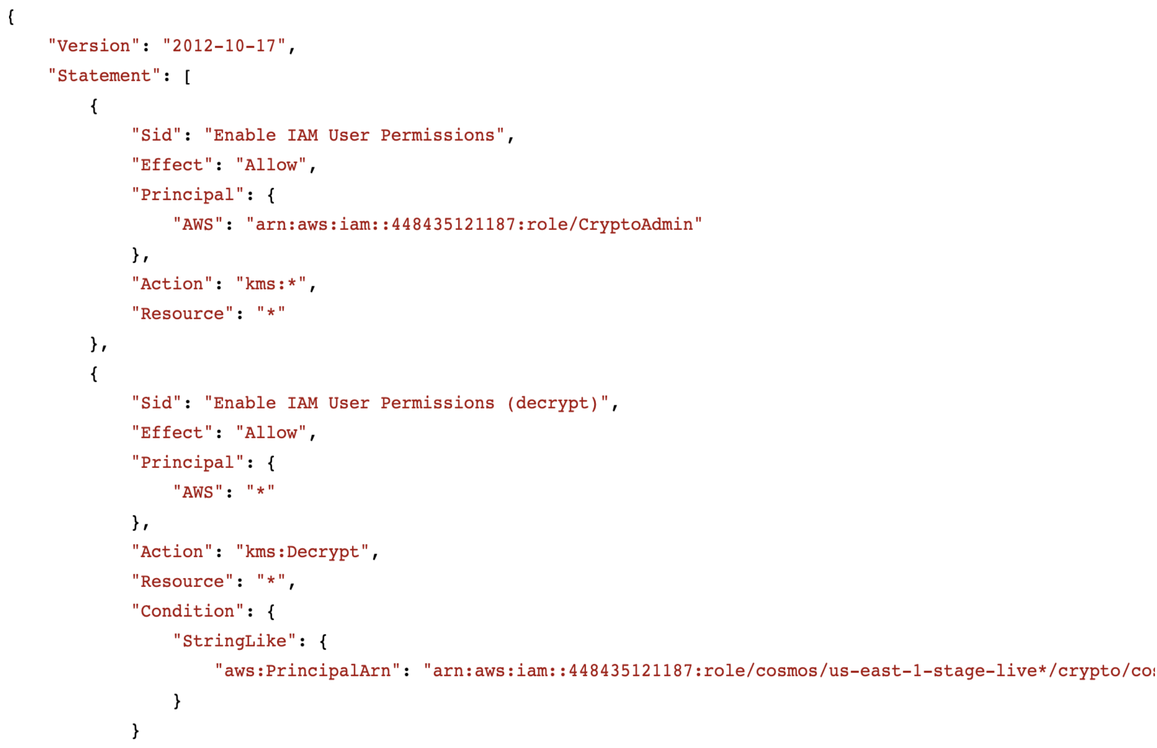
ii. Verify crypto pods have access to the Crypto key in KMS by looking at the key policy. Key policy should have Crypto's IAM role regex, else the pods won't be able to Decrypt using this key



Filter keys by properties or tags — 3 matches

crypto ✕ — Clear filter

	Aliases	Key ID	Status	Key type	Key spec	Key u
☐	test-crypto-garima	75a523c9...	Enabled	Symmetric	SYMMETR...	Encry
☐	crypto-us-stage-live	mrk-0bac...	Enabled	Symmetric	SYMMETR...	Encry
☐	crypto-us-stage-sandbox	mrk-779b...	Enabled	Symmetric	SYMMETR...	Encry



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::448435121187:role/CryptoAdmin"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions (decrypt)",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "kms:Decrypt",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:PrincipalArn": "arn:aws:iam::448435121187:role/cosmos/us-east-1-stage-live*/crypto/co
                }
            }
```

b. Make sure KMS key has not been deleted/marked for deletion (AWS Console should show this info)

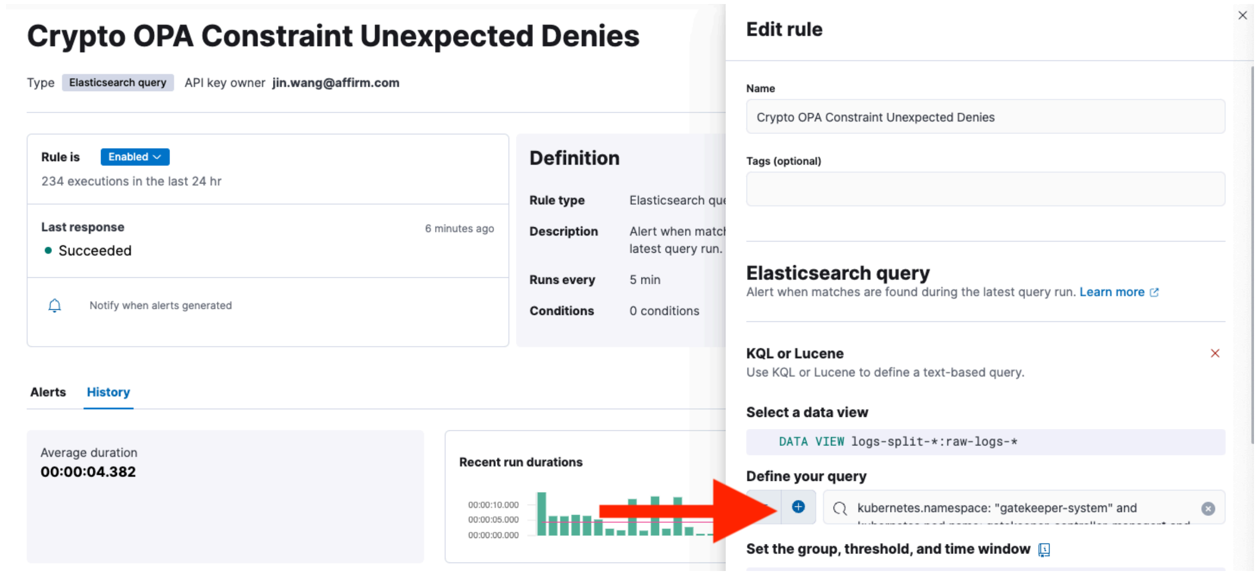
c. Check if S3 Crypto keys have not been updated accidentally to use a different KMS key (Look at the update timestamp on S3 to check when the keys were last rotated)

Crypto OPA Constraint Unexpected Denies

Alerts are set up in Kibana and sent to PD:

- Stage alert
- Prod alert

Search Kibana raw logs using the query in the Kibana alert rule:



Crypto OPA Constraint Unexpected Denies

Type Elasticsearch query API key owner jin.wang@affirm.com

Rule is Enabled
234 executions in the last 24 hr

Last response 6 minutes ago
● Succeeded

Notify when alerts generated

Definition

Rule type Elasticsearch qu
Description Alert when matc latest query run.
Runs every 5 min
Conditions 0 conditions

Alerts History

Average duration
00:00:04.382

Recent run durations

Edit rule

Name
Crypto OPA Constraint Unexpected Denies

Tags (optional)

Elasticsearch query
Alert when matches are found during the latest query run. Learn more

KQL or Lucene
Use KQL or Lucene to define a text-based query.

Select a data view
DATA VIEW logs-split-*:raw-logs-*

Define your query
kubernetes.namespace: "gatekeeper-system" and

Set the group, threshold, and time window

You should see message(s) like this:

```
"denied admission: DELETE operation on Pod resource with name
crypto-rds-cert-bundle-28240560-97rdb is denied for user with groups
[\"system:serviceaccounts\", \"system:serviceaccounts:kube-system\",
\"system:authenticated\"] and username
system:serviceaccount:kube-system:pod-garbage-collector. Access to crypto
namespace is restricted to cryptoadmin role, [\"ec2-coordinator\",
\"system:serviceaccount:cosmos-system:cosmos-dt-ctrl\",
\"system:serviceaccount:keda:keda-operator\",
\"system:serviceaccount:kube-system:horizontal-pod-autoscaler\",
\"system:serviceaccount:kube-system:job-controller\",
\"system:serviceaccount:kube-system:replicaset-controller\"] usernames, or
crypto worker nodes"
```

In most cases, you'll just need to add the username to the allowed username list in the OPA constraint template here and deploy cosmops. Depending on the username, you may want to consult with Cosmos on whether we should allow the username or not.

Some denies are expected and will not be alerted on:

- if the role is `oneloginadmin` or `onelogindev` stage
- if the username is `system:serviceaccount:default:cosmos-cluster-admin-privileged` in prod

If there is an alert for a deny that is expected, update the Kibana rule query to not alert for that role/username.

(The alert is set up in Kibana instead of Chronosphere because OPA gatekeeper did not have the metric we needed. More info here)

Crypto RPC high latency

One possible reason for high latency is high CPU usage and crypto pods failing to scale up quickly enough. Check CPU Usage by pod (% of CPU request) and Pod Count. To manually scale up pods, check HPA Replica Exhausted runbook.

Crypto RPC client call high error percentage

One possible reason for high latency is high CPU usage and crypto pods failing to scale up quickly enough. Check CPU Usage by pod (% of CPU request) and Pod Count. To manually scale up pods, check HPA Replica Exhausted runbook.

Temporarily allowing all egress during deploy / rollout restart

We have limited egress through a combination of network policies and VPC endpoints. However, there may still be some edge cases where some IPs that should be allowed are still blocked. If this happens during a deploy / rollout restart that needs to be done urgently, you can temporarily allow all egress using steps below.

Ensure you have "CryptoAdmin" AWS profile

```
echo "apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: crypto-temp-allow-all-egress-all-pods
  namespace: crypto
spec:
  egress:
  - {}
  podSelector: {}
  policyTypes:
  - Egress" > crypto-netpol.yaml

kubectl apply -f crypto-netpol.yaml
```

# after finishing rollout restart crypto pods / crypto deploy

```
kubectl multicluster -f us-east-1-prod-live -n crypto delete netpol
crypto-temp-allow-all-egress-all-pods
```

Checking blocked IPs due to network policy egress

(only possible in envs where cilium id enabled)

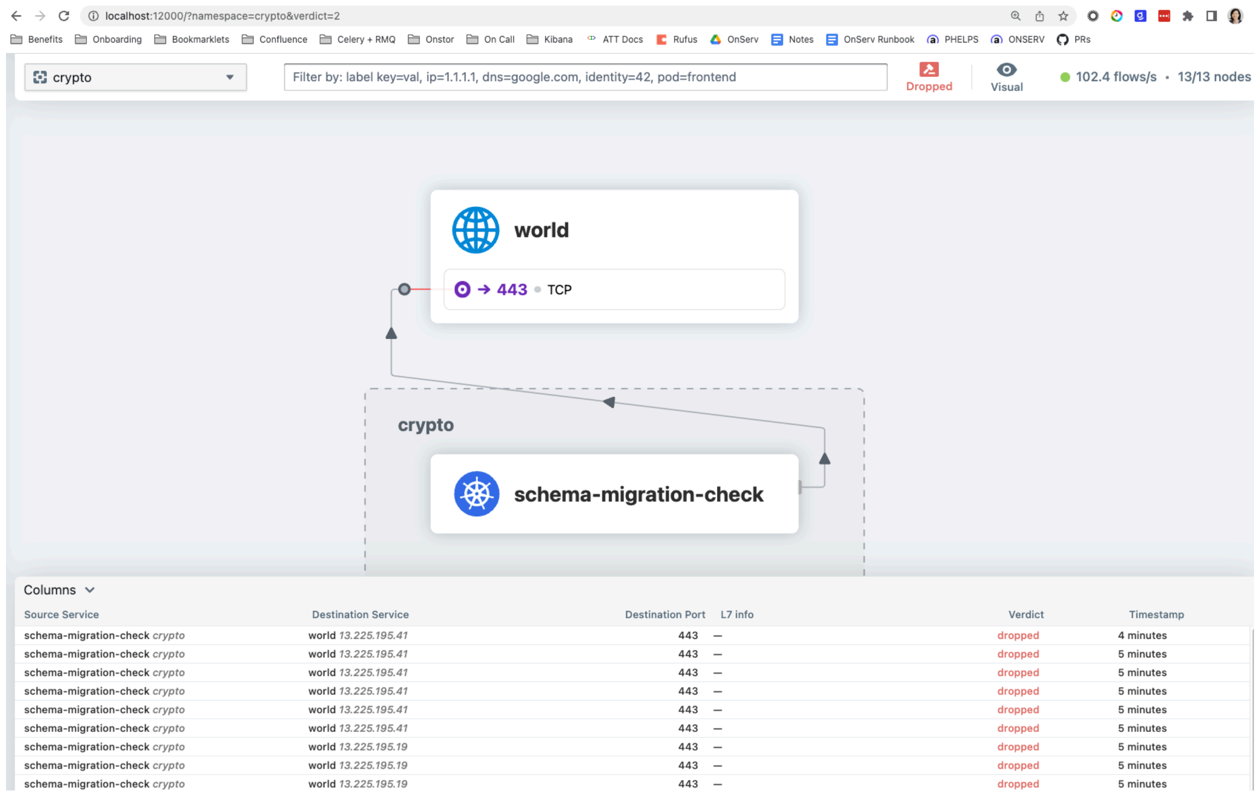
To see which requests are being dropped by cilium:

Install cilium CLI
https://docs.cilium.io/en/stable/gettingstarted/k8s-install-default/#install-the-cilium-cli (may need to remove extra backslashes from the curl command copied from the link)

Run `cilium hubble ui`

Inspect requests w/ Hubble UI https://docs.cilium.io/en/latest/gettingstarted/hubble/

Example UI:



Source Service	Destination Service	Destination Port	L7 info	Verdict	Timestamp
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	4 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.19*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.19*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.19*	443	—	dropped	5 minutes

Crypto Key (DEK) Rotation

Crypto has first-class support for live crypto key rotation. This means that for (non-raw) encrypt/decrypt operations, we can cut over to encrypting with a new key without downtime, and decryptions will use whichever key the ciphertext being decrypted was encrypted with. The key rotation procedure is as follows:

```
# log into AWS account via OneLogin
# activate crypto venv

mankey admin repl # opens REPL

rotate # select which keys to rotate
# this will create new, inactive versions of the keys and upload them to S3

# wait 12 minutes for inactive keys to be loaded in crypto pods

activate_keys
# this activates keys by removing `decrypt_only` flag on key data

upload
# this uploads activated keys to S3

# activated keys will be loaded in crypto pods
```

QA scripts

See Crypto: Prod Readiness

Decrypto runbook

More info about Decrypto here

Decrypt Crypto ARI Request (Decrypto)

Users can use the Decrypto tool in Mordor to submit a request. This should work in all prod and stage envs. Here is the URL for US prod live:
https://www.affirm.com/mordor/debugapp/decrypto/

They will need to input their PGP key, which they can get from https://keyserver.team.affirm.com/ by inputting their email, clicking on the link, and copying the PGP key, which will look something like:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
                           .......
-----END PGP PUBLIC KEY BLOCK-----
```

In prod, legal will need to approve the request. In stage, anyone with the `decrypto_admin` role can approve the request. Everyone on Onserv should have access to this role.

After a period of time, the status of the request should change to “Done” in Decrypto in Mordor, and there should be a decrypted data file link that the user can click on to get the decrypted data. If the decrypted data file is empty, then the crypto ARI was not found.

Airflow DecryptoApprovalRequestTask Failed

Check task logs. There should be a link to the logs in the alert

To see a list of all task runs that have failed, check Airflow UI:
https://airflow2.prod-live-jobs.affirm-keyhole.com/dagrun/list/?_flt_0_dag_id=adhoc_run&_flt_2_run_id=DecryptoApprovalRequestTask&_flt_0_state=failed

You can also check logs in Kibana. If the task failed to start, the logs may not show up in Kibana, but you can still find them in the Airflow UI.

Currently there are no common failure cases, but here is some general info that might help with debugging:

- Task is kicked off here
- Task is defined here
- Task uses adhoc_run DAG

Cert-manager runbook

Cert-manager dashboard

https://affirm.chronosphere.io/dashboards/d/cert-manager/cert-manager?orgId=1

Expired certificate / Certificate unable to renew

Describe the certificate resource to see what the error is and follow the steps for the specific error

```
kubectl describe -n <namespace> cert <cert_name>
```

The certificate request has failed to complete and will be retried: failed to retrieve provisioner

To resolve this, delete the CertificateRequest resource associated with the Certificate resource and trigger a manual renewal using cmctl. First, look at the CertificateRequest resources in the namespace and there should be one that has a ready state of “False”. If you describe this resource it will show the status as “failed to retrieve provisioner”. Delete this CertificateRequest resource and then retrigger a renewal on the Certificate.

```
# Locate CertificateRequest resources with a "False" ready status
kubectl get certificaterequests -n <namespace> -o json | jq -r '.items[] |
select(.status.conditions[] | select(.type=="Ready" and .status=="False"))'

# Delete the CertificateRequest resource with the "False" ready status
kubectl delete certificaterequests -n <namespace> <certificaterequest_name>

# Install cmctl if you don't already have it
brew install cmctl

# Manually trigger cert-manager renewal on the certificate
cmctl renew -n <namespace> <cert_name>
```

Boba PKI runbook

Boba PKI containers failing to start up

Unable to issue cert error

```
Message:  time="2023-10-28T09:12:57Z" level=error msg="unable to issue cert" err="aws login
request send failed: Post \"https://vault.prod.aff/v1/auth/aws/login\": x509: certificate is
not valid for any names, but wanted to match vault.prod.aff" tag=vault_err
```

If you see this error in the containers, there may be an issue with Vault. Reach out to Infra Eng in #ask-infrastructure-engineering. Infa Eng owns Vault, but if you’d like to do some troubleshooting, check Vault Troubleshooting.

Vault Troubleshooting

Check if any new Vault instances were added recently in AWS console:

- Link for US prod

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:instanceState=running;search=:vault;v=3;$case=tags:true%5C,client:false;$regex=tags:false%5C,client:false;sort=desc:launchTime

Check if the common name in the certificate is correct by teleporting into Vault instance and running:

```
openssl x509 -in /etc/pki/tls/certs/localhost.crt -noout -text
```

If only some of the instances are misconfigured and some instances are configured correctly, deregister the bad instances from that Vault NLB target group (It's expected that half the instances are unhealthy because vault have active / passive setup, and passive instances are expected to have "unhealthy" status)

- Link for US prod live

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroup:targetGroupArn=arn:aws:elasticloadbalancing:us-east-1:906324658258:targetgroup/vault/b709e7f671351d51

Cilium Runbook

Cilium Nuances and Things to Know

Cilium Prod Readiness Review

Cilium Dashboard

https://affirm.chronosphere.io/v3/dashboards/cilium-dashboard/cilium-dashboard?orgId=1

Install Cilium and Hubble CLI

Install Cilium CLI:
https://docs.cilium.io/en/stable/gettingstarted/k8s-install-default/#install-the-cilium-cli

Install Hubble CLI:
https://docs.cilium.io/en/v1.13/gettingstarted/hubble_setup/#install-the-hubble-client

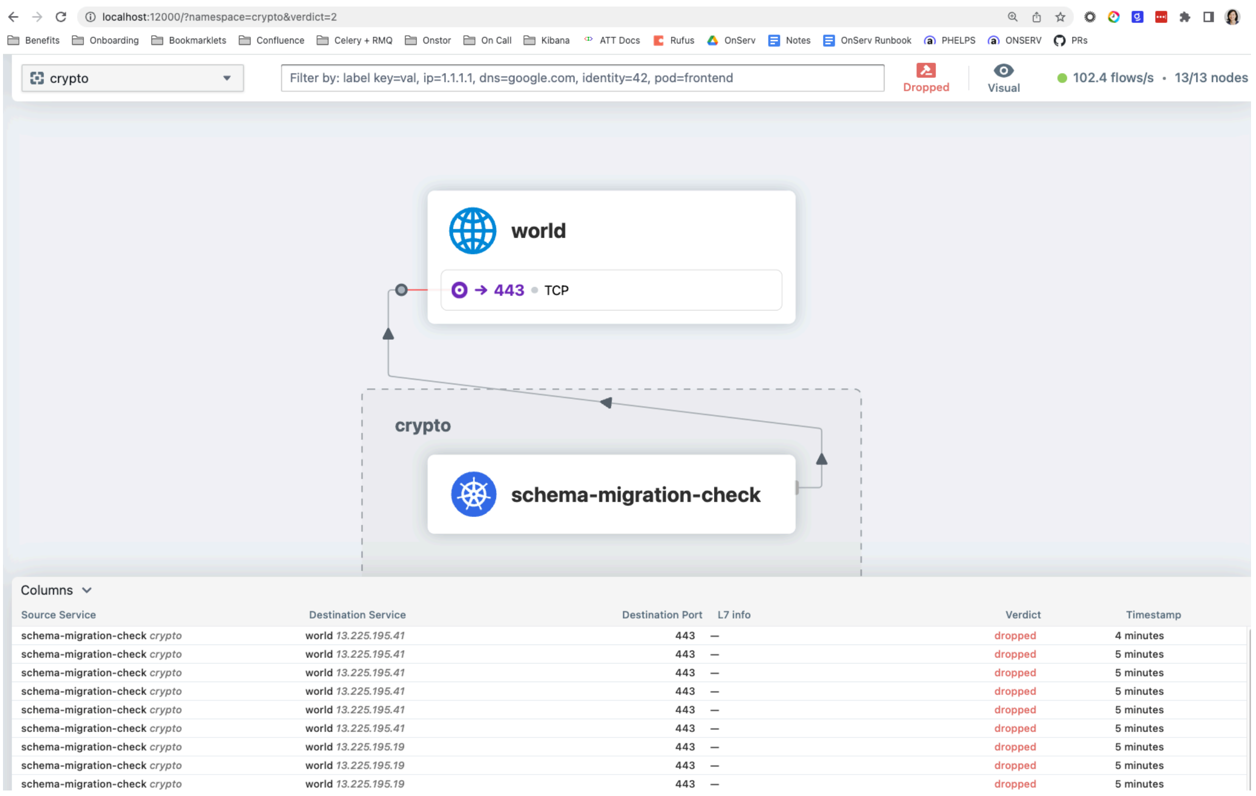
Viewing Dropped Requests by Cilium via NetworkPolicy

Using Hubble UI

SCC into cluster and run `cilium hubble ui`

A browser window should open where you can select the namespace you want to monitor.
Inspect requests w/ Hubble UI https://docs.cilium.io/en/latest/gettingstarted/hubble/

Example UI:



Source Service	Destination Service	Destination Port	L7 info	Verdict	Timestamp
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	4 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.41*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.19*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.19*	443	—	dropped	5 minutes
schema-migration-check *crypto*	world *13.225.195.19*	443	—	dropped	5 minutes

Using Hubble CLI

To run any hubble CLI commands, **you first need to run `cilium hubble port-forward` in a terminal window**. This will set up the appropriate port-forwarding of the hubble relay port to the local machine. Leave this terminal window open and open another one to run hubble commands.

The main command to run is `hubble observe` to view all flows within a cluster. See `hubble observe --help` to check what flags you can add to narrow down flows.

Example: `hubble observe -f -n monolith -t policy-verdict --verdict DROPPED` to view dropped network policy verdicts for flows related to the monolith namespace.

Retrieve Cilium Pod Managing a Particular K8s Pod

```
curl -sLO https://raw.githubusercontent.com/cilium/cilium/main/contrib/k8s/k8s-get-cilium-pod.sh
chmod +x k8s-get-cilium-pod.sh
./k8s-get-cilium-pod.sh <POD> <NAMESPACE>
```

List Unmanaged K8s Pods

```
curl -sLO https://raw.githubusercontent.com/cilium/cilium/main/contrib/k8s/k8s-unmanaged.sh
chmod +x k8s-unmanaged.sh
./k8s-unmanaged.sh
```

Pods that show up in this list will not be managed by Cilium and won’t have network policies enforced properly that have been applied to ingress to or egress from these pods. No application pods should show up in this list but some core daemonset pods or karpenter pods may show up which is fine since there are no network policies on those pods. Read more about this behavior here Cilium Managed/Unmanaged Pod(s) Testing .

Other Useful Guides

- Cosmos Runbook