#### 



# Adding Observability to the Managed Tokens Service

Shreyas Bhat Scientific Computing Monitoring Workshop July 24, 2024

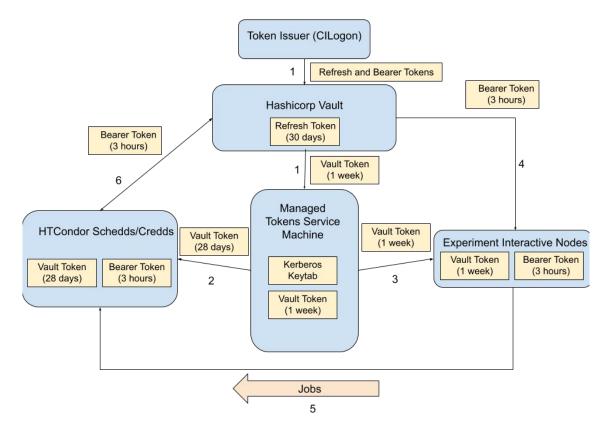


#### What is the Managed Tokens Service?

- Store production vault tokens in condor *credds*
- Push production vault tokens to interactive nodes, keep them refreshed
- Written in Go (performant, easy/safe concurrency, avoid Python "dependency hell")



#### Where the service fits in



Fermilab

#### **Executables of Managed Tokens Service**

#### • token-push:

- Stores vault token in HTCondor credd for experiment/role combination and push copies of vault token to interactive nodes
- Notifies experiment stakeholders and operators if there is any error
- Runs every hour.

#### • refresh-uids-from-ferry:

- Queries FERRY to pull down the applicable UIDs for the configured UNIX accounts
- Runs daily each morning

#### run-onboarding-managed-tokens:

- Onboards a new experiment or experiment account to the Managed Tokens Service
- Run as needed



#### Normal operation of token-push

- 1. Uses a kerberos service principal to authenticate to the Hashicorp vault server
- 2. Sets appropriate credkey in HTGETTOKENOPTS based on kerberos principal  $\rightarrow$  Lets condor\_vault\_storer/htgettoken know where the refresh token is in the vault
- 3. condor\_vault\_storer
  - a. Resets the refresh token in the vault and obtains a vault token
  - b. Stores this vault token in the HTCondor *credd*
- 4. Pushes this vault token to experiment interactive nodes
- 5. Notifies experiment stakeholders and/or operators if there is any error in any of the previous steps



#### **Components to keep track of**

- All of the operations on the last slide plus:
- Configuration parsing
- Local sqllite database reads/writes
- Notification generation/sending
- Most of these operations (with exception of condor\_vault\_storer) are done concurrently
- How do we monitor all of this?

### Logs with Loki

- Standard logs at /var/log/managed-tokens on managed tokens machine
  - One regular log, one debug log per executable
- But searching through those can be tedious: Send logs to Loki
- Queryable log store FE via LogQL:
  - o {app="managed-tokens",command="token-push"} | json |
    experiment="gm2" AND role="production"
- Simply required adding a configured Loki hook to existing logging library INFO and higher-level logs are sent to Loki



### Logs with Loki (2)

	Split 🔡 Add to dashboard	⊙ Last 1 hour CDT × Q 📿 Run query × ▷ L						
~ A (Loki)		0000						
Log browser > {app="managed-tokens",command="token-push"}   json   experiment="gm2" AND role="production"								
Query type         Range         Instant         Line limit         O         auto         Resolution         1/1         ~								
+ Add query S Query history I Inspector								
Log volume								
40 20								
0 16:40 16:45 16:50 16:55 17:00 17:05 17:10 17:15 - info	17:20 17:25	17:30 17:35						
Logs								
Time 💽 Unique labels 🜑 Wrap lines 🂽 Prettify JSON 🌑 Dedup None Exact Numbers Signature		Display results Newest first Oldest first						
Common labels: managed-tokens token-push production gm2 info production managed-tokens Line limit: 1000 (25 returned) Total bytes processed: 315 kB								
<pre>&gt; 2824-87-19 16:58:54 {"destinationFilename":"/tmp/jobsub_default_role_gm2_45651","experiment":"gm2","level":"info","msg":"Success copying file to destination", "node":"gm2gpvm88", "role":"production", "sourceFilename":"/tmp/managed_toke ns_default_role_file_2686599999","time":"2024-87-19T16:58:54 ("destinationFilename":"/tmp/tu45651","experiment":"gm2","level":"info","msg":"Success copying file to destination", "node":"gm2gpvm88","role":"production", "sourceFilename":"/var/lib/managed_tokens/service-cred d-vault-tokens/vt_u47535-jobsub81.fnal.gov-gm2_production","time":"2024-87-19T16:58:54-85:08"} &gt; 2024-87-19 16:58:53 ("destinationFilename":"/tmp/vt_u45651","experiment":"gm2","level":"info","msg":"Success copying file to destination","node":"gm2gpvm88","role":"production","sourceFilename":"/var/lib/managed-token s/service-credt-vault-tokens/vt_u47535-jobsub81.fnal.gov-gm2_production","time":"2024-87-19T16:58:53-85:08"} &gt; 2024-87-19 16:58:53 ("destinationFilename":"/var/lib/managed-token s/service-credt-vault-tokens/vt_u47535-jobsub81.fnal.gov-gm2_production","time":"2024-87-19T16:58:53-85:08"} &gt; 2024-87-19 16:58:53 ("destinationFilename":"/tmp/tpudzedfault_role_gm2_45651","experiment":"gm2","level":"info","msg":"Success copying file to destination","node":"gm2gpvm88","role":"production","sourceFilename":"/var/lib/managed-token s/service-credt-vault-tokens/vt_u47535-jobsub81.fnal.gov-gm2_production","time":"2024-87-19T16:58:53-85:08"} &gt; 2024-87-19 16:58:53 ("destinationFilename":"/tmp/tpudzedfault_role_gm2_45651","experiment":"gm2","level":"info","msg":"Success copying file to destination","node":"gm2gpvm87","role":"production","sourceFilename":"/var/lib/managed_token ns_default_role_file_2686599999","time":"2024-87-19T16:58:53-85:08"}</pre>								

**Fermilab** 

#### **Metrics**

- Prometheus metrics pushed each run to a Landscape pushgateway
- Standard prometheus setup is for web services; for batch processes like this, need to use <u>pushgateway</u>: extra webserver that can receive metrics, then serves them to normal Prometheus server
- Metrics
  - managed\_tokens\_stage\_duration\_seconds: Per executable, per stage (setup, processing, cleanup). How long each stage took to run.
  - managed\_tokens\_last\_ferry\_refresh: Timestamp of when refresh-uids-from-ferry executable last got information from FERRY.
  - managed\_tokens\_failed\_services\_push\_count: Count of how many services registered a failure to push a vault token to a node in the current run of token-push. Basically, a failure count.
  - managed\_tokens\_last\_token\_push\_timestamp: Timestamp of when token-push last pushed a particular service vault token to a particular node.
  - Managed\_tokens\_last\_token\_store\_timestamp: Timestamp of when token-push last stored a particular service vault token in a particular credd.

🛠 Fermilab

 $\circ \quad \text{And more!} \quad$ 



#### **Metrics (2)**

Prometheus Alerts Graph Status - Help Classic UI	* ( )
□ Use local time □ Enable query history 🕑 Enable autocomplete 🕑 Enable highlighting 🕑 Enable linter	
<pre>Q managed_tokens_last_token_store_timestamp{experiment="dune"}</pre>	S Execute
Table Graph	Load time: 267ms Resolution: 14s Result series: 7
< Evaluation time >	
managed_tokens_last_token_store_timestamp{credd="dunegpschedd01.fnal.gov", experiment="dune", job="managed_tokens", role="production", service="dune_production"}	1721429902.3104067
managed_tokens_last_token_store_timestamp{credd="dunegpschedd02.fnal.gov", experiment="dune", job="managed_tokens", role="production", service="dune_production"}	1721429902.7990355
managed_tokens_last_token_store_timestamp{credd="jobsub01.fnal.gov", experiment="dune", job="managed_tokens", role="production", service="dune_production"}	1721429912.426958
managed_tokens_last_token_store_timestamp{credd="jobsub02.fnal.gov", experiment="dune", job="managed_tokens", role="production", service="dune_production"}	1721429912.896547
managed_tokens_last_token_store_timestamp{credd="jobsub03.fnal.gov", experiment="dune", job="managed_tokens", role="production", service="dune_production"}	1721429913.3713026
managed_tokens_last_token_store_timestamp{credd="jobsub04.fnal.gov", experiment="dune", job="managed_tokens", role="production", service="dune_production"}	1721429913.8435159
managed_tokens_last_token_store_timestamp{credd="jobsub05.fnal.gov", experiment="dune", job="managed_tokens", role="production", service="dune_production"}	1721429914.3057816

**Remove Panel** 



## Tracing

- While metrics and logs give great insight into issues that crop up, traces help visualize WHERE in the program flow something went wrong
- Serve as a visual index rather than grepping through the logs
- Then, for more details, you can go to the logs if needed

- We added OpenTelemetry tracing that gets sent to Landscape Jaeger Tracing collector
- For example: <u>https://landscape.fnal.gov/jaeger/trace/0eee23d1cfd43126e6dc3d2d184f9007</u>



## Tracing (2)

←	← ✓ <u>managed-tokens: token-push</u> 0			sh 0eee23d		Find	⑦	~ ~ X	ℜ Trace Timeline ∽
Trace Start July 19 2024, 21:58:01.698 Duration 1m 6s Ομε			024, 21:58:01.698   Duration 1m 6s	Services 1   Depth 11   Total Spans 4004		33.15s	49.72s		1m 6s
Servic	e & O	peratio	$\sim \rightarrow \approx \gg$	0µs	16.57s		33.15s	49.72s	1m 6s
	>	mana	ged-tokens worker.PushTokensWorker				922.99ms		
	>	mana	ged-tokens worker.PushTokensWorker				819.67ms	•	
	~	manag	ed-tokens worker.PushTokensWorker_a				1.02s	-	
		∼ ma	naged-tokens worker.PushTokensWork				894ms	-	
_		>	managed-tokens worker.PushToken				323.85ms	•	
		>	managed-tokens worker.PushToken				473.04ms	<b>a</b> [1]	
		~	Imanaged-tokens worker.pushToN				341.03ms	s 🖣	
				worker.pushToNode			Service: managed-tok	ens   Duration: 34	1.03ms   Start Time: 48.75s
		<ul> <li>Tags: destinationFilename = /tmp/jobsub_default_role_hypot-fife-test_24362 error = true internal.span.format = jaeger node = fermicloud848 otel.st</li> <li>Process: deployment.environment = production otel.library.name = managed-tokens</li> </ul>					otel.status_code = ERR		
				> Logs (1)					
								S	panID: 905d7c27db28d798 🕒
			✓ managed-tokens fileCopier.Copy				340.33ms	3	
			> <b>0</b> managed-tokens fileCo				340.33ms	5	
	>	mana	ged-tokens worker.PushTokensWorker				3.26s		
	>	manag	ged-tokens worker.PushTokensWorker				4.74s		

**‡** Fermilab

### **Tracing to visualize concurrency (3)**

#### Trace Start July 19 2024, 21:58:01.698 | Duration 1m 6s | Services 1 | Depth 11 | Total Spans 4004

0µs	16.57s	33.15s		49.72s	1m 6s		
	_						
Service & Operation $\lor$ > $\Leftrightarrow$ »	0µs	16.57s	33.15s	49.72s	1m 6s		
managed-tokens notifications.registerNot	1µs						
✓ managed-tokens notifications.ServiceEm	l 1µs						
	notifications.ServiceEmailManager.runServiceNotificationHandler       Service: managed-tokens       Duration: 1µs       Start Time: 228.91ms         > Tags: internal.span.format = jaeger       service = hypot-fife-test_production       service: managed-tokens       service: managed-tokens         > Process:       deployment.environment = production       otel.library.name = managed-tokens       service: managed-tokens         SpanID:       ed594c797765a7eb       ca						
> managed-tokens notifications.Servi	notifications.ServiceEmailManager.runServiceNotificationHandler_anonFunc Tags: internal.span.format = jaeger Process: deployment.environment = production otel.library.name = managed-tokens			Service: managed-tokens   Duration: 1m 6s   Start Time: 228.92ms SpanID: 85bbc87712c03a40 (9)			
	322.67ms 217.78ms						

We can see here that a *hypot-fife-test\_production* ServiceNotificationHandler is launched concurrently and stays alive for almost the entire run - waiting for messages, like the one the previous slide's error would have generated - you can SEE the concurrency!

🛟 Fermilab

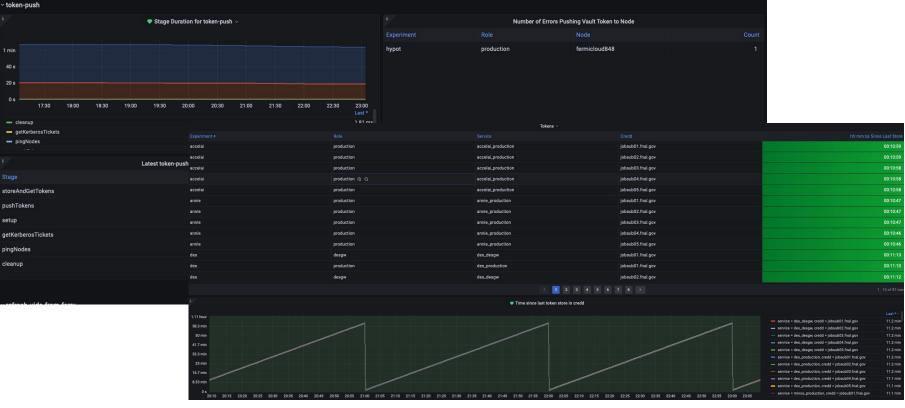
#### **Dashboards**

- Fifemon dashboards:
  - Last token push by service/node: <u>https://fifemon.fnal.gov/monitor/d/QOV9N\_ilz/managed-tokens-service-time-sin</u> <u>ce-last-token-push?orgId=1</u>
  - Last token store in credd by service/credd (with alerts after 3 hrs of failure): <u>https://fifemon.fnal.gov/monitor/d/w4mAD\_ilk/managed-tokens-service-time-sin</u> <u>ce-credd-store</u>
  - Service Health Dashboard: <u>https://fifemon.fnal.gov/monitor/d/bGDwH9mSz/managed-tokens-service-servic</u> <u>e-health</u>



### **Dashboards (2)**





#### **Fermilab**

## Alerting

- Use dashboards to send alerts to #fifealerts Slack channel
  - Time since a token was stored in condor *credd* > 3 hrs
  - Any stage from *refresh-uids-from-ferry* takes > 5 min
  - Time since UIDs refreshed from FERRY > 2 days
- Also, original notifications from *token-push* itself:
  - Emails to stakeholders, FIFE
  - Abridged version sent to #fifealerts



#### **Future**

- Pretty much done no major work left in adding observability
- TODO: Export traces to new Landscape OpenTelemetry collector rather than current Landscape Jaeger collector
  - One function change, since library APIs between Jaeger/OpenTelemetry are nearly-identical
  - Then rest of tracing code in *managed tokens* should seamlessly plug-in



#### Summary

- Managed tokens service executables are multi-threaded with numerous concurrent operations running at any time
- Adding observability has helped us pinpoint issues very quickly when they occur:
  - *Alerting/dashboards* fed by *metrics* tells us that something went wrong
  - *Traces* show us very quickly in which components of which executable the issue occurred
  - Logs sent to Loki (INFO level) let us quickly find the exact operation within the relevant component that went wrong

🛠 Fermilab

- Debug logs on the deployment machine can help us look at details of faulty operation
- All of these components working together has saved a LOT of troubleshooting time



#### Resources

- Github repo: <u>https://github.com/fermitools/managed-tokens</u>
- Wiki page:
  - https://fifewiki.fnal.gov/wiki/Managed Tokens Service
- Loki: <u>https://grafana.com/docs/loki/latest/</u>
- Prometheus: <u>https://prometheus.io/</u>
- Prometheus Pushgateway: <u>https://prometheus.io/docs/practices/pushing/</u>
- Using Metrics in Grafana:
   <u>https://prometheus.io/docs/visualization/grafana/</u>
- OpenTelemetry Tracing: <u>https://opentelemetry.io/docs/</u>



# Thank you!



#### **Extra Slides**



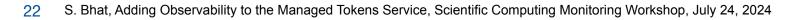
### **Loki Configuration with logrus**

244	<pre>// Loki. Example here taken from README: https://github.com/YuKitsune/lokirus/blob/main/README.md</pre>						
245	<pre>loki0pts := lokirus.NewLokiHook0ptions().</pre>						
246	<pre>// Grafana doesn't have a "panic" level, but it does have a "critical" level</pre>						
247	<pre>// https://grafana.com/docs/grafana/latest/explore/logs-integration/</pre>						
248	<pre>WithLevelMap(lokirus.LevelMap{log.PanicLevel: "critical"}).</pre>						
249	WithFormatter(&log.JSONFormatter{}).						
250	WithStaticLabels(lokirus.Labels{						
251	"app": "managed-tokens",						
252	"command": currentExecutable,						
253	"environment": devEnvironmentLabel,	Code to add loki hook to					
254	})	logrus logging library					
255	<pre>lokiHook := lokirus.NewLokiHookWithOpts(</pre>						
256	<pre>viper.GetString("loki.host"),</pre>						
257	lokiOpts,						
258	log.InfoLevel,	Link to code on Github					
259	log.WarnLevel,						
260	log.ErrorLevel,						
261	log.FatalLevel)						
262							
263	log.AddHook(lokiHook)						

Configuration for loki in managed tokens config file



**Fermilab** 



#### **Prometheus configuration**

- Configuration to push metrics to pushgateway at http://fifelog.fnal.gov:9091
- Set up MetricsRegistry using <u>prometheus Go client</u> <u>library</u> and <u>prometheus Go pushgateway library</u>
- Set up metrics in each executable, like <u>this</u> (link to Github)



#### **Tracing configuration**

- Configuration to export all traces to <u>https://landscape.fnal.gov/jaeger-collector/api/traces</u> : note
   there is now an OpenTelemetry-compatible trace collector that should be used instead
- Set up TracerProvider to export traces (example)
- Set up executables to use this TracerProvider when creating new spans (<u>see here</u>)
- In each operation/function, start a new span (<u>example</u>)

