Presto Training Series, Session 3: Securing Presto

Try Presto: www.prestosql.io

Dain Sundstrom and Manfred Moser

26 August 2020

Today's Speakers

Manfred Moser

Developer, author,
and trainer at Starburst

Manfred is an open source developer and advocate. He is co-author of the book Presto: The Definitive Guide, and a seasoned trainer and conference presenter. He has trained over 20,000 developers for companies such as Walmart Labs, Sonatype, and Telus.

Dain Sundstrom

Co-creator of Presto and
CTO at Starburst

Dain is a co-creator of Presto, co-founder of the Presto Software Foundation, and CTO at Starburst. Prior to Starburst, Dain was a Software Engineer at Facebook, A Software Architect at Proofpoint, founded the Apache Geronimo project, and was one of the original JBoss authors.

Agenda

- Presto overview / review
- HTTPS setup
- Presto authentication, including password & LDAP authentication
- Authorization to access your data sources
- Five minute break
- Secure communication in the cluster
- Secrets usage for configuration files including catalogs
- Securing Hive connector
- Q&A

Questions

- Ask any time
- Use the meeting **Questions** feature
- Manfred screens, collects and interjects
- Dedicated Q&A in break and at end

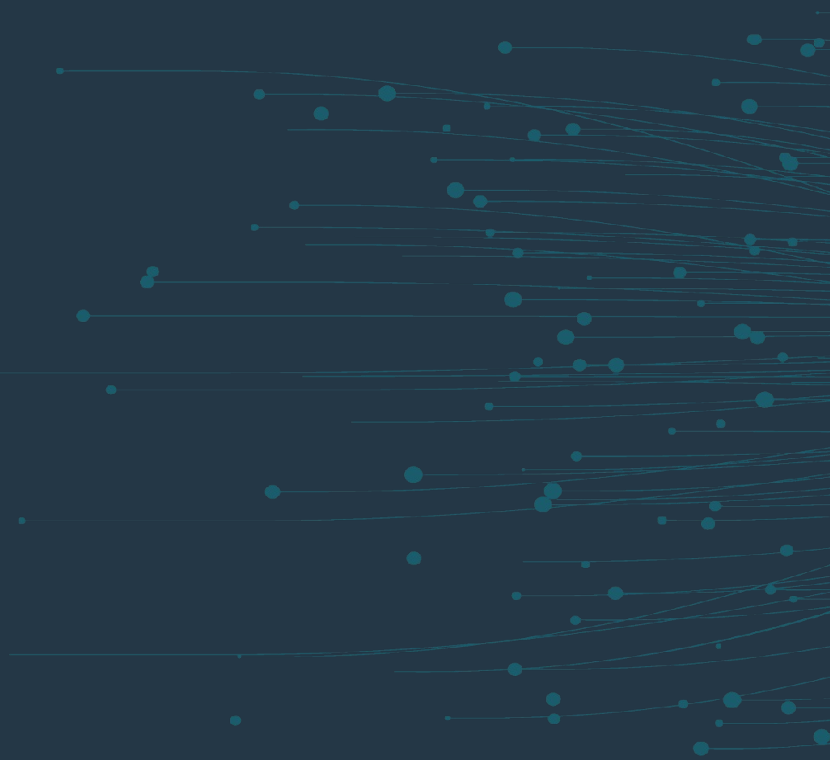
Starburst

Some advice for attendees

- This is a fast-paced overview – don't try to follow along during class
- Instead focus and pay attention – security is always a tricky topic
- Use the demo video after class to test with Presto and CLI
- Learn at your own pace
- Use video recording and slides from class as reference to learn more
- Apply skills for your own use case

Presto overview

… probably just a recap for you



What is Presto?

High performance ANSI SQL engine

- SQL support for any connected data source - SQL-on-anything
- Cost-based query optimizer
- Proven horizontal scalability

Open source project

- Very active, large community
- User driven development
- Huge variety of users
- Prestosql.io

Separation of compute and storage

- Scale query processing and data sources independently
- Query storage directly
- No ETL or data integration necessary

Presto everywhere

- No cloud vendor lock-in
- No storage engine vendor lock-in
- No Hadoop distro vendor lock-in
- No database lock in

Why use Presto?

Fastest time-to-insight

- High performance query processing
- Low barrier of entry for users
- Massive scalabilty
- High concurrency
- Direct access to storage

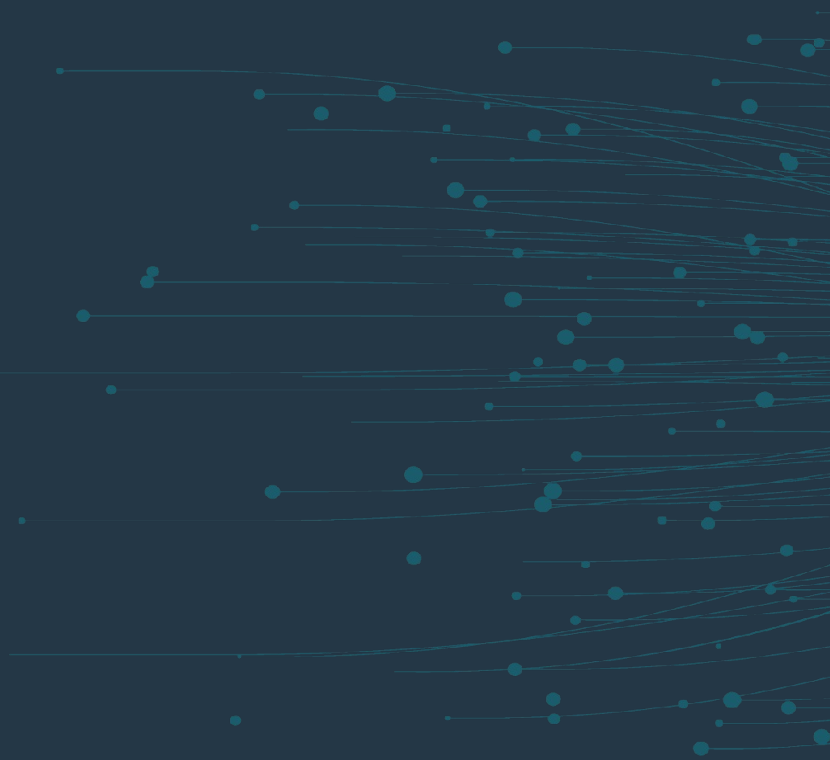
Lower cost

- Reduced need to copy and move data
- Avoid complex data processing
- Scale storage and compute independently
- Only run computes when processing queries
- One data consumption layer

Avoid data lock in

- No more data silos, departmental copies
- Query data with the existing skills and tools - SQL + BI tools
- Query any data source
- Move data
- Create optionality

Presto security



Tips and notes

- We will focus on securing a Presto cluster
- You may need help from your corporate security team
- Testing command line tools:
  - openssl
  - curl
  - htpasswd
- Other tools
  - Text editor of your choice
  - Browser of your choice

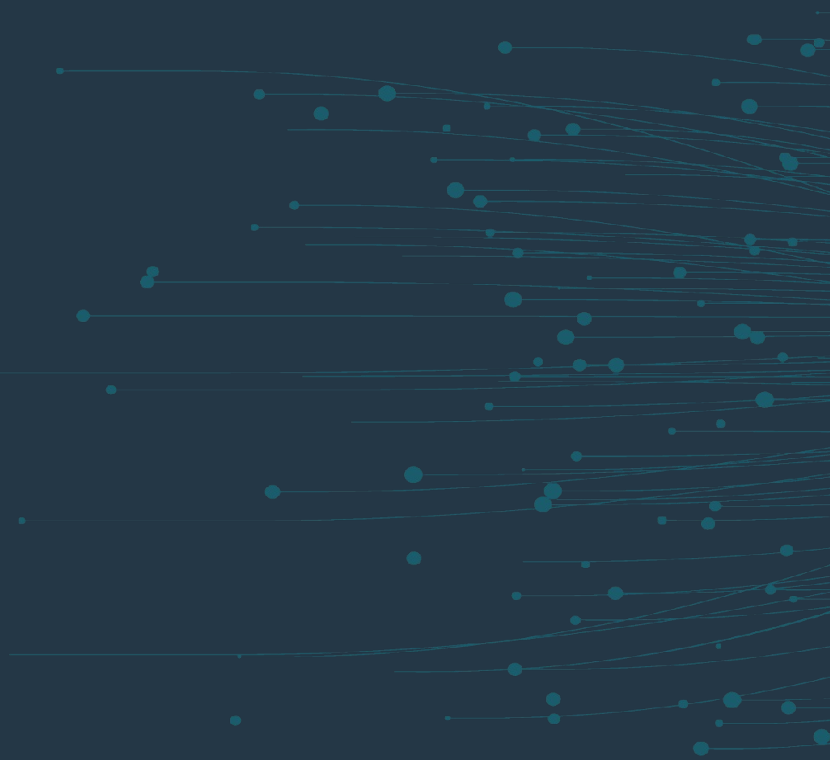
Process for securing Presto

- Follow the process in order
- Don’t skip steps
- Do one step at a time, don’t combine steps
- Restart server after each change
- Verify after each step

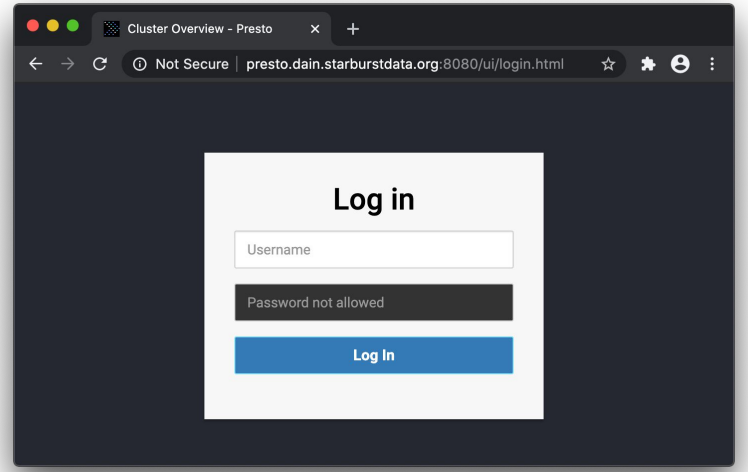
What do we need to secure

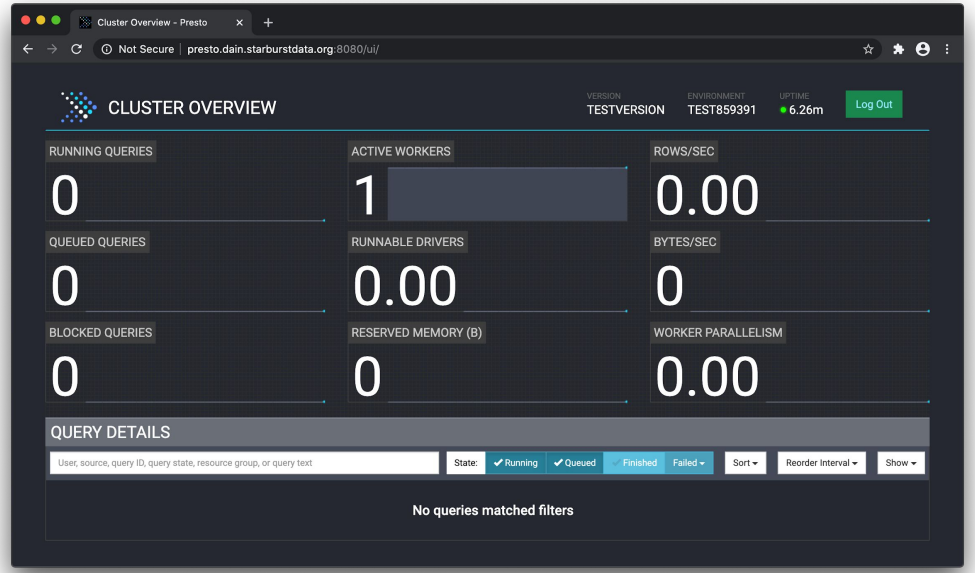
- Clients to cluster
  - Encryption - secure communication - prevent eavesdropping
  - Authentication - who are you?
  - Authorization - are you allowed to do that?
- Inside the cluster
  - Secure communication between Presto servers
  - Secure secrets in configuration files
- Cluster to data sources
  - Securing connection to Hive (Metastore, HDFS, S3, GCS, ADLS)

Insecure Presto



Verify HTTP with the Web UI





Starburst

Verify HTTP with the CLI

```
> presto --server http://presto.dain.starburstdata.org:8080
presto> select 'rocks' as presto;
 presto
--------
 rocks
(1 row)
```

CLI failures

Bad server name:

```
> presto --server http://unknown:8080

presto> select 'bad host' as example;

Error running command: java.net.UnknownHostException: unknown: nodename nor servname
provided, or not known
```

Bad port:

```
> presto --server http://presto.dain.starburstdata.org:9999

presto> select 'bad port' as example;

Error running command: java.net.ConnectException: Failed to connect to
presto.dain.starburstdata.org/127.0.0.1:9999
```

Part 1: Client to server

Encryption

Enable HTTPS

Starburst

Approaches

1. HTTPS proxy or load balancer (LB)
   - Normally the easiest approach (well understood by corporate security)
   - Built into all cloud providers, and systems like K8s
   - Proxy or LB handles HTTPS details, and minimal config on Presto side

2. Add SSL/TLS certificate to Presto coordinator
   - Can be more difficult to get certificates from corporate security
   - Presto handles HTTPS details directly

HTTPS proxy or load balancer (LB)

- Hook into existing setup at your company
  - In cloud environments, a load balancer is typically already setup with trusted certificate
  - In K8s, use an Nginx ingest controller
- Load balancer terminates SSL/TLS and connects to Presto coordinator using HTTP
  - All Presto servers are contained in a virtual private network
  - LB adds request headers with original client information:

```
X-Forwarded-Proto: https
X-Forwarded-Host: presto.dain.starburstdata.org
Forwarded: for=192.0.2.60;proto=https;by=203.0.113.43
```

- Enable processing of Forwarded headers in `<presto>/etc/config.properties`

```
http-server.process-forwarded=true
```

Starburst

Add the SSL/TLS certificate to the Presto coordinator

- Get an SSL/TLS certificate and private key for your server
  - Ideally the certificate is globally trusted by browsers
- Certificate will be in PEM or JKS format
  - Presto supports PEM (PKCS #8 and PKCS #1) directly
  - JKS is a Java specific format and is also supported

Inspect the PEM file

- PEM is plain text
- Should contain a certificate and private key sections. For example:

```
-----BEGIN CERTIFICATE-----
MIIFHjCCAwYCCQCmg7+WHUP8ADANBgkqhkiG9w0BAQsFADBRMQswCQYDVQQGEwJV
…
-----END CERTIFICATE-----
-----BEGIN PRIVATE KEY-----
MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQDTWPax1rfWIvK/
...
-----END PRIVATE KEY-----
```

- If provided as two separate files, just concatenate into one file.
- If it says ENCRYPTED PRIVATE KEY, then you will need a password.

Verify the PEM file certificate

- Dump certificate contents

```
> openssl x509 -in your-pem-file -text -noout
```

- Browsers are now enforcing a 398 days is the maximum length

```
Validity
    Not Before: Aug 23 18:43:07 2020 GMT
    Not After : Nov 21 18:43:07 2020 GMT
```

- Browsers now require subject alternative name

```
X509v3 Subject Alternative Name:
    DNS:presto.dain.starburstdata.org
```

- Legacy common name mostly ignored, but good to have

```
Subject: CN=presto.dain.starburstdata.org
```

Verify the PEM file private key

```
> openssl rsa -in your-pem-file -check -noout
RSA key ok
```

- Will prompt for password if encrypted
- This only works with RSA keys

Verify the JKS file

- Dump JKS file contents

```
> keytool -list -v -keystore your-jks-file
```

- Verify file contains a private key

```
Entry type: PrivateKeyEntry
```

- Verify validity

```
Valid from: Thu Mar 17 09:40:46 PDT 2016
     until: Wed Mar 17 09:40:46 PDT 2021
```

- Verify subject alternative name

```
SubjectAlternativeName [
  DNSName: presto.dain.starburstdata.org
]
```

Starburst

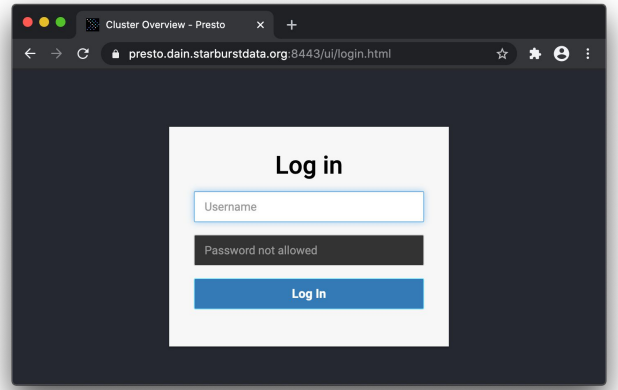
Configure Presto

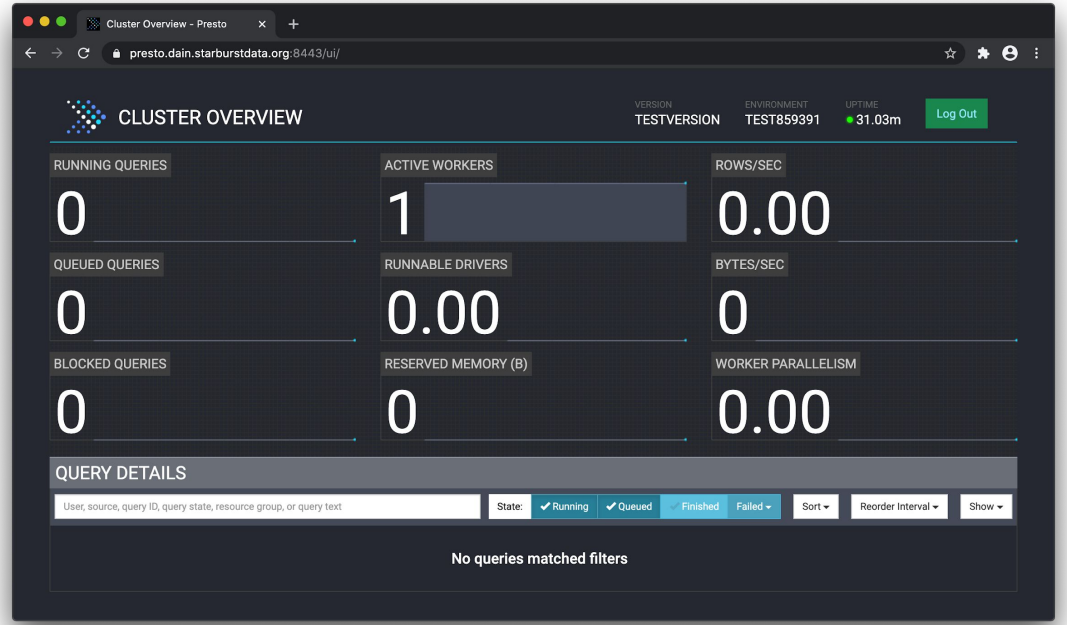
- On Presto coordinator set configuration:

```
http-server.https.enabled=true
http-server.https.port=8443
http-server.https.keystore.path=your-pem-or-jks-file
http-server.https.keystore.key=keystore_password
```

- Restart server

Verify HTTPS with the Web UI





Starburst

Verify HTTP with the CLI

```
> presto --server https://presto.dain.starburstdata.org:8443
presto> select 'rocks' as presto;
 presto
--------
 rocks
(1 row)
```

CLI Failure: Not trusted

```
> presto --server https://presto.dain.starburstdata.org:8443

presto> select 'rocks' as presto;

Error running command: javax.net.ssl.SSLHandshakeException: PKIX path building
failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target
```

Add --truststore-path:

```
> presto --server https://presto.dain.starburstdata.org:8443 \
    --truststore-path server.pem

presto> select 'rocks' as presto;

 presto
--------
 rocks
(1 row)
```

Starburst

CLI truststore password problems

Bad password:

```
> presto --server https://presto.dain.starburstdata.org:8443 \
     --truststore-path server.p12 --truststore-password invalid

presto> select 'rocks' as presto;

Error running command: Error setting up SSL: keystore password was incorrect
```

Missing password:

```
> presto --server https://presto.dain.starburstdata.org:8443 \
    --truststore-path server.p12

presto> select 'rocks' as presto;

Error running command: javax.net.ssl.SSLException: Unexpected error:
java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be
non-empty
```

CLI --insecure

Allow ANY certificate! Only use this for testing:

```
> presto --server https://presto.dain.starburstdata.org:8443 --insecure
presto> select 'rocks' as presto;
 presto
--------
 rocks
(1 row)
```

Starburst

Presto authentication

Overview

- Supported authentication:
  - Username and password with file and LDAP
  - Kerberos
  - Client certificate
  - JSON Web Token
- There is a plugin in interface for username and password

Password file authentication

Password file authentication

- Enable password authentication in `<presto>/etc/config.properties`

  ```
  http-server.authentication.type=PASSWORD
  ```

- Password authenticator is configured in `<presto>/etc/password-authenticator.properties`

  ```
  password-authenticator.name=file
  file.password-file=/path/to/password.db
  file.refresh-period=1m
  file.auth-token-cache.max-size=1000
  ```

Create a password file

- Password file is created with the `htpasswd` utility from the Apache HTTP Server.
- Create an empty password file:

```
> touch password.db
```

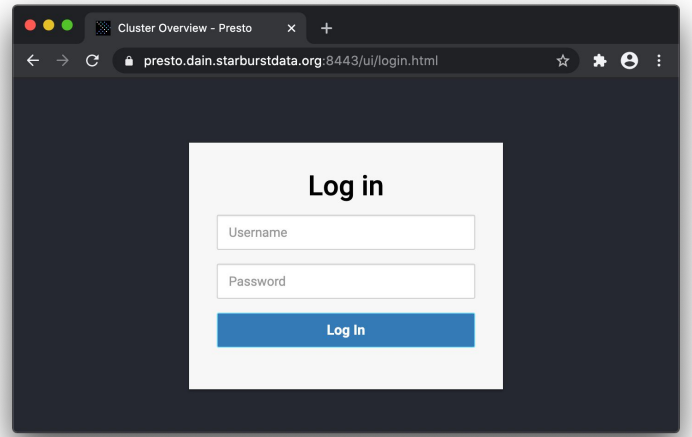
- Add or update password for user dain:

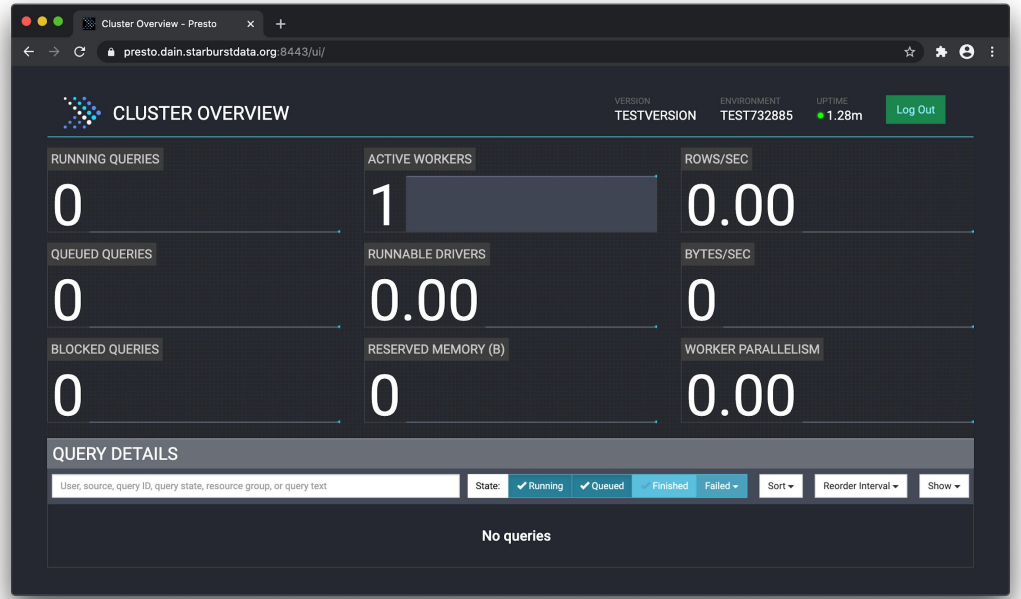
```
> htpasswd -B -C 10 password.db dain
New password:
Re-type new password:
Adding password for user dain
```

- Verify password for user dain:

```
> htpasswd -v password.db dain
Enter password:
Password for user dain correct.
```

Verify HTTPS with the Web UI





Starburst

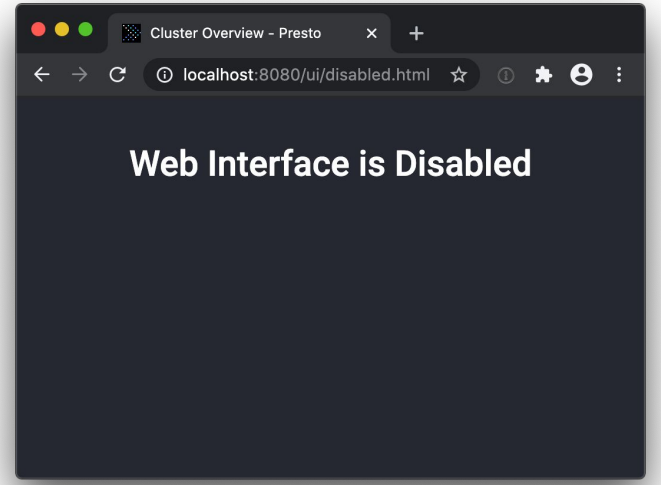
Verify HTTPS with the CLI

```
> presto --server https://presto.dain.starburstdata.org:8443 --password
Password:
presto> select 'rocks' as presto;
 presto
--------
 rocks
(1 row)


> presto --server https://presto.dain.starburstdata.org:8443 --user dain --password
```

Starburst

HTTP is automatically disabled




```
> presto --server http://presto.dain.starburstdata.org:8080
presto> select 'test';
Error running command: Error starting query
```

Starburst

Common problems

Invalid username or password

```
> presto --server https://presto.dain.starburstdata.org:8443 --password
Password:
presto> select 'bad password' as example;
Error running command: Authentication failed: Access Denied: Invalid credentials
```

No password: missing --password flag

```
> presto --server https://presto.dain.starburstdata.org:8443
presto> select 'no password' as example;
Error running command: Authentication failed: Unauthorized
```

LDAP authentication

LDAP authentication

Enable password authentication in `<presto>/etc/config.properties`

```
http-server.authentication.type=PASSWORD
```

Password authenticator is configured in `<presto>/etc/password-authenticator.properties`

```
password-authenticator.name=ldap
ldap.url=ldaps://ldap-server:636
ldap.ssl-trust-certificate=<PEM-file>
```

- Authentication styles:
  - User bind pattern
  - User bind pattern and validation LDAP query
  - Service user with LDAP query

User bind pattern

1. Connect to LDAP with interpolated username and password

Configuration:

```
ldap.user-bind-pattern=<validation-query>
```

Active Directory Example:

```
${USER}@corp.example.com
```

OpenLDAP Example:

```
${USER},OU=America,DC=corp,DC=example,DC=com
```

User bind pattern with LDAP query

1. Connect to LDAP with interpolated username and password
2. Execute LDAP query to validate user (e.g., a group membership check)

Configuration:

```
ldap.group-auth-pattern=<validation-query>
```

Active Directory Example:

```
(&(objectClass=person)(sAMAccountName=${USER})
(memberof=CN=AuthorizedGroup,OU=US,DC=corp,DC=example,DC=com))
```

OpenLDAP Example:

```
(&(objectClass=inetOrgPerson)(uid=${USER})
(memberof=CN=AuthorizedGroup,OU=US,DC=corp,DC=example,DC=com))
```

Service user with LDAP query

1. Connect as fixed service user (instead of authenticating user)
2. Execute LDAP query to extract LDAP distinguished name for user
3. Connect using extracted LDAP distinguished name and password

Configuration:

```
ldap.bind-dn=<service-user-dn>
ldap.bind-password=<service-user-password>
ldap.group-auth-pattern=<query-for-user-dn>
```

Verify LDAP!

- Same as password file -

Kerberos authentication

Kerberos - Don’t do it

Kerberos is very difficult to work with:

- Inscrutable error messages
- Misleading error messages
- Built into OS and JVM (can not to use better implementation)
- Kerberos servers (KDC) are not very reliable
- Most organizations with Kerberos have LDAP
- Doesn’t work well with browsers

Other authentication options

Client certificate authentication

- Client authenticates with an SSL/TLS client certificate
- Server will allow any client certificate trusted by the truststore

Configuration:

```
http-server.authentication.type=certificate
http-server.https.truststore.path=<path-to-pem-or-jks>
http-server.https.truststore.key=<truststore-password>
```

CLI arguments:

```
--keystore-path=<path-client-pem-or-jks>
--keystore-password=<keystore-password>
```

JSON Web Token authentication

- Very basic JWT implementation
- JWT must be signed by a key on the local file system (no JWK)
- Key can be selected based on `${KID}`
- Helpful for service accounts

Configuration:

```
http-server.authentication.type=jwt
http-server.authentication.jwt.key-file=<pem-or-hmac>
http-server.authentication.jwt.required-issuer=<optional>
http-server.authentication.jwt.required-audience=<optional>
```

CLI arguments:

```
--access-token <token>
```

Multiple authenticators

- Multiple authenticators can be used at the same time
- Only one or each type (so no LDAP and Password File)
- Helpful for migrating between systems
- Web UI only supports a single authenticator
- Each authenticator is checked in order

Configuration:

```
http-server.authentication.type=PASSWORD,KERBEROS
```

User mapping

- Some authenticator result in *complex* usernames
  - Kerberos: `alice@example.com`
  - Client Certificate: `CN=alice,OU=Finance,O=Acme,C=US`
- User mapping extracts username with a regular expressions
- If pattern no pattern matches, authentication is denied

Simple configuration:

```
http-server.authentication.<auth>.user-mapping.pattern=
    (.*)@example.com
```

Verify:

```
SELECT current_user;
```

Starburst

Complex user mapping

Configuration:

```
http-server.authentication.<auth>.user-mapping.file=<file>
```

Example mapping:

```
{
    "rules": [
        {
            "pattern": "test@example\\.com",
            "allow": false
        },
        {
            "pattern": "(.+)@example\\.com"
        },
        {
            "pattern": "(?<user>.+)@(?<region>.+)\\.example\\.com",
            "user": "${user}_${region}"
        }
    ]
}
```

Starburst

Authorization

Authorization (a.k.a. access control)

- Check if user is allowed to take specific actions
- Presto has two levels of authorization
  - System: global plugin that is called for all actions
  - Connector: catalog level checks for a single datasource
- Built in system authorization systems:
  - default: allow all except user impersonation
  - allow-all: allow everything
  - read-only: write and alter operations are not allowed
  - file: rules in a file determine allowed operations

Configuration `<presto>/etc/access-control.properties`

```
access-control.name=read-only
```

File based system access control

Configuration `<presto>/etc/access-control.properties`

```
access-control.name=file
security.config-file=<rules-json-file>
security.refresh-period=1m
```

Rules JSON file

- Sections for different checks (e.g., query, catalog, impersonation)
- Each rule is checked top down, and first match decides
- If no rules match, access is denied
- Check docs for default when no rules are provided

Query rules

```
{
  "queries": [
    {
      "user": "admin_.*",
      "allow": ["execute", "kill", "view"]
    },
    {
      "user": "alice",
      "allow": ["execute", "kill"]
    },
    {
      "allow": ["execute"]
    }
  ]
}
```

Note: Users always have permission to view or kill their own queries

Catalog rules

```
{
  "catalogs": [
    {
      "user": "admin",
      "catalog": "(mysql|system)",
      "allow": "all"
    },
    {
      "catalog": "hive",
      "allow": "all"
    },
    {
      "user": "alice",
      "catalog": "postgresql",
      "allow": "read-only"
    }
  ]
}
```

Note: By default users can access system catalog unless there is a rule

Schema rules

```
{
  "schemas": [
    {
      "user": "admin",
      "owner": true
    },
    {
      "user": "test",
      "schema": "test",
      "owner": false
    },
    {
      "schema": "default",
      "owner": true
    }
  ]
}
```

Starburst

Table rules

```
{
  "tables": [
    {
      "user": "admin",
      "privileges": ["SELECT", "INSERT", "DELETE", "OWNERSHIP"]
    },
    {
      "user": "banned_user",
      "privileges": []
    },
    {
      "schema": "default",
      "table": ".*",
      "privileges": ["SELECT"]
    }
  ]
}
```

Starburst

All together

```
{
  "queries": [...],
  "catalogs": [...],
  "schemas": [...],
  "tables": [...],
}
```

Put all rules in one file as a JSON list.

Verify

Set the rules file to:

```
{
    "catalogs": [
        {
            "catalog": "system",
            "allow": "none"
        }
    ]
}
```

Run query:

```
presto> select * from system.runtime.nodes;

Query 20200824_183358_00000_c62aw failed: Access Denied: Cannot access catalog
system
```

Starburst

Client to server summary

Process

- Enable HTTPS
  - If possible, use a load balancer or proxy
  - If possible, get a globally trusted certificate
- Enable authentication
  - Start with password file
  - Switch to preferred system
  - Avoid Kerberos!
- Enable authorization
  - Use file based rules to control global security

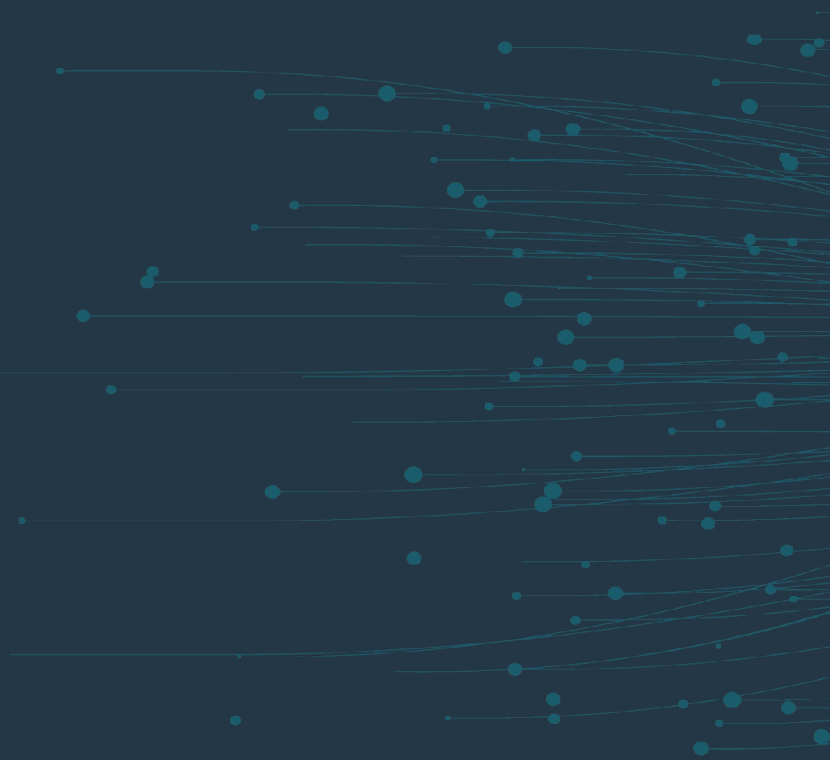
5 minute break

And if you stick around:

- Browse prestosql.io
- Join us on Slack
- Submit questions
- Star github.com/prestosql/presto

Part 2: Internal security and connector security

Internal security



Starburst

Securing the Presto cluster itself

- Presto version ≥ 337
- Secure internal communications
  - Shared secret
  - Optionally enable internal HTTPS
- Secrets management
- Management endpoints

Secure internal communications

Shared secret

- Presto servers authenticate to each other using a shared secret
- Shared secret must be set to secure Presto servers
- All servers must have same value
- Generate secret using: `openssl rand 512 | base64`

Configure `<presto>/etc/config.properties`

```
internal-communication.shared-secret=<secret>
```

Internal HTTPS

- Only enable internal HTTPS if needed
- We recommend disabling HTTP entirely in this case
- Ever server **must** have a certificate that matches **internal** name
- Internal name can be IP, HOSTNAME, or FQDN (or explicitly set)

Configuration `<presto>/etc/config.properties` (all machines)

```
http-server.http.enabled=false
discovery.uri=https://<coordinator fqdn>:<https port>
internal-communication.https.required=true
node.internal-address-source=FQDN
internal-communication.https.truststore.path=<pem-or-jks>
internal-communication.https.truststore.key=<password>
```

Secrets management

Secrets in configuration files

- Load password and secrets from process environment
- Simply replace the secret part with `${ENV:VARIABLE}`
- Supported by most provisioning systems (e.g., Ansible, K8s)

Example:

```
internal-communication.shared-secret=${ENV:SECRET}
```

AWS secret manager to ENV:

```
aws secretsmanager get-secret-value \
  --secret-id ${1} \
  --query SecretString --output text | \
  jq -r '.secrets[] | "export " + .name + "=\"" + .value + "\""'
```

Starburst

Management endpoints

Secure management endpoints

- Management REST endpoints
  - `/v1/info/state:` Graceful server shutdown
  - `/v1/jmx:` Read management stats
  - `/v1/node:` Read node info
- Management endpoints require authentication and authorization
  - Authentication can be any supported system
  - File-based system access control:

```
{
    "system_information": [
        {
            "user": "dain",
            "allow": ["read", "write"]
        }
    ]
}
```

Starburst

Management examples

```
> curl -u dain https://presto.dain.starburstdata.org:8443/v1/node
Enter host password for user 'dain':
[]

> curl https://dain:password@presto.dain.starburstdata.org:8443/v1/node
[]

> curl https://presto.dain.starburstdata.org:8443/v1/node
Unauthorized

> curl https://dain:invalid@presto.dain.starburstdata.org:8443/v1/node
Access Denied: Invalid credentials
```

Starburst

Hive catalog security

Securing a Hive catalog

- Hive authorization (table level security)
- Metastore authentication
- Storage authentication
  - HDFS
  - S3
  - GCP
  - Azure

Hive catalog authorization

- Supported authorization
  - legacy (default value): see docs
  - read-only: same as system level
  - file: same as system level
  - sql-standard: use permissions stored in Hive Metastore
- SQL standard is the most common
  - If using Hive roles, check the docs

Configure in catalog properties file:

```
hive.security=sql-standard
```

Metastore authentication

- Authenticate Presto process to Hive metastore
- Hive metastore only support Kerberos, so only do this if required
- Easiest solution is to check existing `hive-site.xml`

```
<property>
  <name>hive.metastore.sasl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hive.metastore.kerberos.keytab.file</name>
  <value>/etc/hive/conf/hive.keytab</value>
</property>
<property>
  <name>hive.metastore.kerberos.principal</name>
  <value>hive/_HOST@YOUR-REALM.COM</value>
</property>
```

Metastore authentication configuration

Example configuration:

```
hive.metastore.authentication.type=KERBEROS
hive.metastore.service.principal=hive/metastore-host.example.com@EXAMPLE.COM
hive.metastore.client.keytab=/etc/hive/conf/hive.keytab
hive.metastore.client.principal=hive/_HOST@YOUR-REALM.COM
```

- The `_HOST` placeholder in client principal is replaced with actual host name (same as Hive)
- Presto can impersonate actual end user if required

```
hive.metastore.thrift.impersonation.enabled=true
```

HDFS authentication

- Authenticate Presto to HDFS servers if necessary
- Similar to metastore, but no easy configs to copy

Example:

```
hive.hdfs.authentication.type=KERBEROS
hive.hdfs.presto.principal=presto@EXAMPLE.COM
hive.hdfs.presto.keytab=/etc/presto/hdfs.keytab
```

- Presto can impersonate actual end user if required

```
hive.hdfs.impersonation.enabled=true
```

Hive Kerberos debugging

- Double check Hive actually works with Kerberos
- Ping KDC from Presto server (KDC name maybe in krb5.conf)
- Enable Kerberos debugging in JVM properties:

  ```
  -Dsun.security.krb5.debug=true
  ```
- Dump keytab

  ```
  klist -k -t <keytab file name>
  ```
- Verify keytab

  ```
  kinit -k -t <keytab file name> <service principal name>
  ```
- Ask for help from Hadoop admins or corporate security

S3 authentication

- Normally EC2 instance has IAM Roles to access S3, so just works
- There are a lot of manual options (see docs)
- Credentials to access S3 can be configured based on rules:

```
hive.s3.security-mapping.config-file=<mapping-rules.json>
hive.s3.security-mapping.refresh-period=1m
```

S3 mapping file

```
{
  "mappings": [
    {
      "prefix": "s3://bucket-name/abc/",
      "iamRole": "arn:aws:iam::123456789101:role/test_path"
    },
    {
      "prefix": "s3://special-bucket/",
      "accessKey": "AKIAxxxaccess",
      "secretKey": "iXbXxxxsecret"
    },
    {
      "user": "test.*",
      "iamRole": "arn:aws:iam::123456789101:role/test_users"
    },
    {
      "iamRole": "arn:aws:iam::123456789101:role/default"
    }
  ]
}
```

S3 extra credentials

Config:

```
hive.s3.security-mapping.iam-role-credential-name=iam_role
hive.s3.security-mapping.colon-replacement=_
```

Mappings File:

```
{
  "mappings": [
    {
      "iamRole": "arn:aws:iam::123456789101:role/test_default",
      "allowedIamRoles": [
        "arn:aws:iam::123456789101:role/test1",
        "arn:aws:iam::123456789101:role/test2"
      ]
    }
  ]
}
```

```
> presto --server https://presto.dain.starburstdata.org:8443
    --extra-credential iam_role=arn_aws_iam__123456789101_role/test1
```

Starburst

GCP authentication

- Use service account JSON key:

  `hive.gcs.json-key-file-path=<gcs-keyfile>`

- GCS key from client extra credentials:

  `hive.gcs.use-access-token=true`

```
> presto --server https://presto.dain.starburstdata.org:8443
    --extra-credential hive.gcs.oauth=<gcs-token>
```

Azure Storage

WASB

```
hive.azure.wasb-storage-account=<account>
hive.azure.wasb-access-key=<key>
```

ADLS Gen2

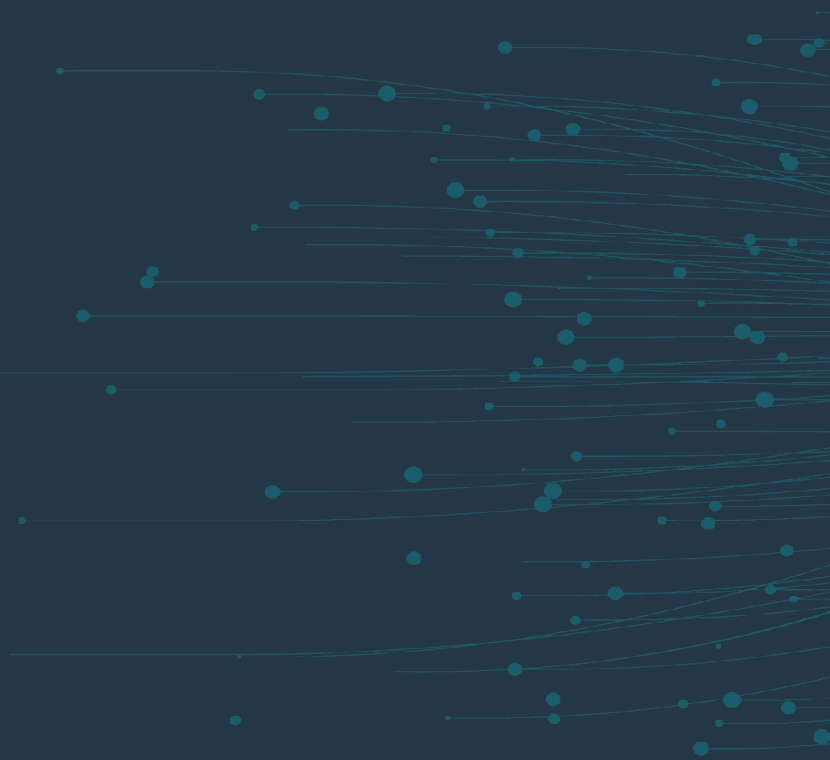
```
hive.azure.abfs-storage-account=<account>
hive.azure.abfs-access-key=<key>
```

ADLS Gen1

```
hive.azure.adl-client-id=<application-id>
hive.azure.adl-credential=<application-secret>
hive.azure.adl-refresh-url=<oauth-endpoint>
```

Starburst security

Help from the experts



Starburst benefits

- Starburst Enterprise Presto
  - many general security related extensions
  - additional security features in connectors
  - LTS with security backports
- Support team
  - years of experience
  - securing Presto
  - tuning Presto and more
- Development team
  - involved in Presto architecture and code from the start

Starburst Enterprise Presto

- Users and groups LDAP access control
- First class Apache Ranger integration
  - any catalog, not just Hive
  - fine-grained policies
  - column masking
  - row-level filtering
  - LDAP user sync
- Okta authorization

- Connector enhancements
  - Kerberos pass through
  - username/password pass through
  - user impersonation
  - IAM credentials

And much more beyond security

Starburst

Wrapping up

Presto Training Series

Join the Presto creators again for more:

- Advanced SQL in Presto with David - recording available
- Presto Query Processing with Martin - recording available
- Configuring and Tuning Presto Performance with Dain (9 Sept)

Starburst

Presto Summit series

Diverse information about Presto and real world usage

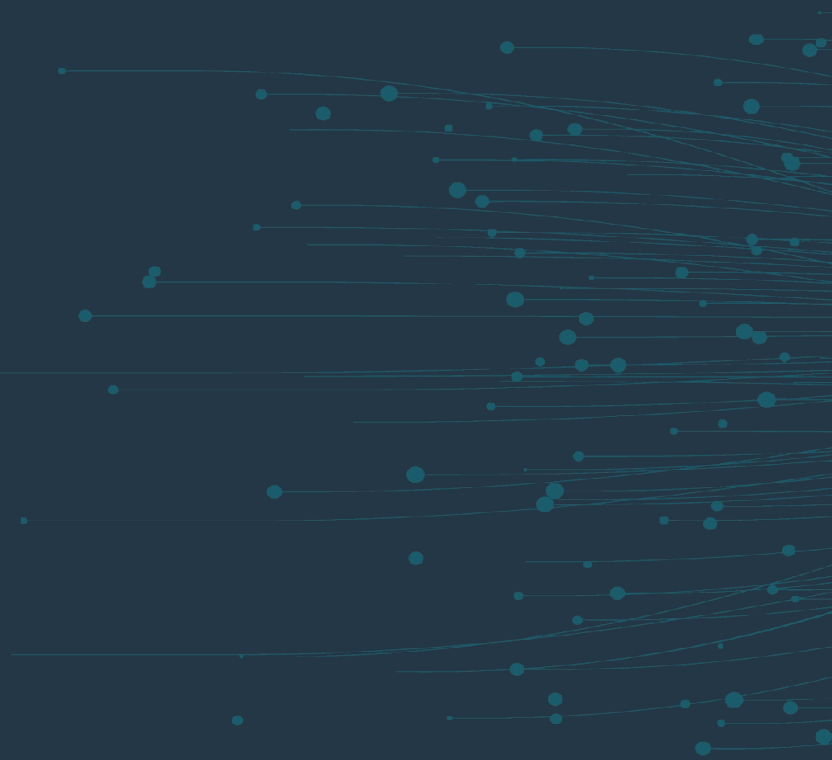
- State of Presto - recording available
- Presto as Query Layer at Zuora - recording available
- Presto Migration at Arm Treasure Data - recording available
- Presto for Analytics at Pinterest - 19 Aug

And finally ...

- Learn more from our website and documentation at prestosql.io
- Join us on slack at prestosql.io/slack
- Get a free digital copy of Presto: The Definitive Guide
- Thank you for hanging out with us
- See you next time

Starburst

Your question
Our answers …



Starburst