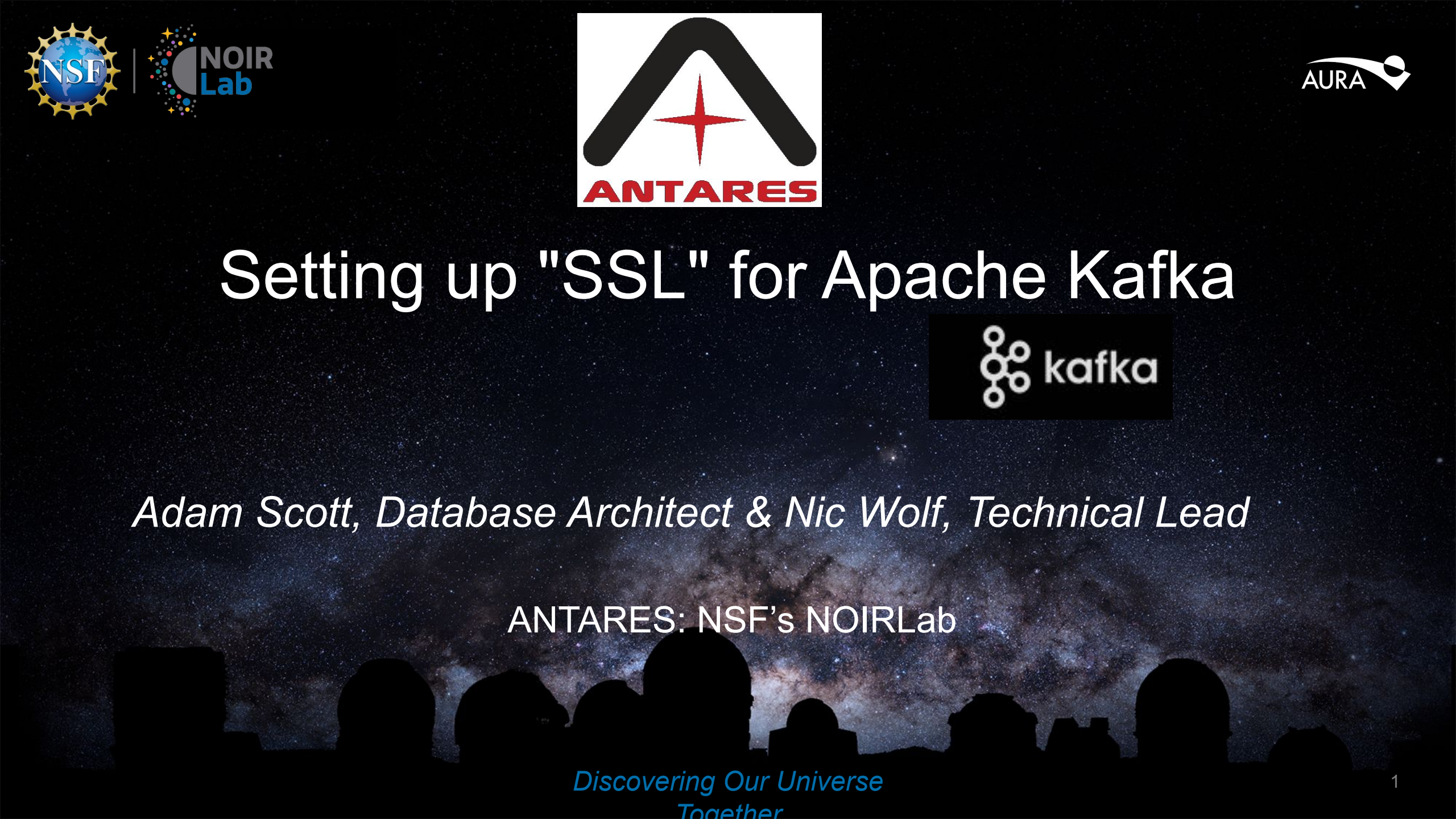


Setting up "SSL" for Apache Kafka

Adam Scott, Database Architect & Nic Wolf, Technical Lead

ANTARES: NSF's NOIRLab

Discovering Our Universe Together

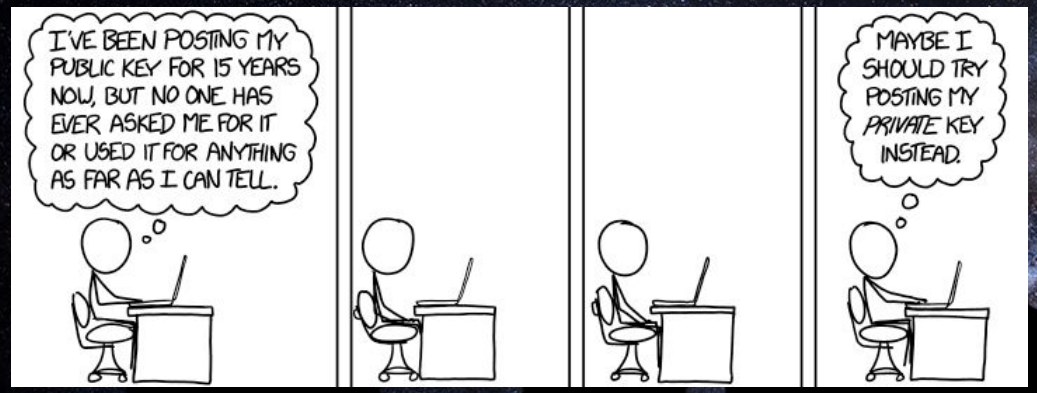
Setting up SSL for Kafka

- What is SSL?
- Why setup SSL?
- How to setup SSL Certificates

kafka

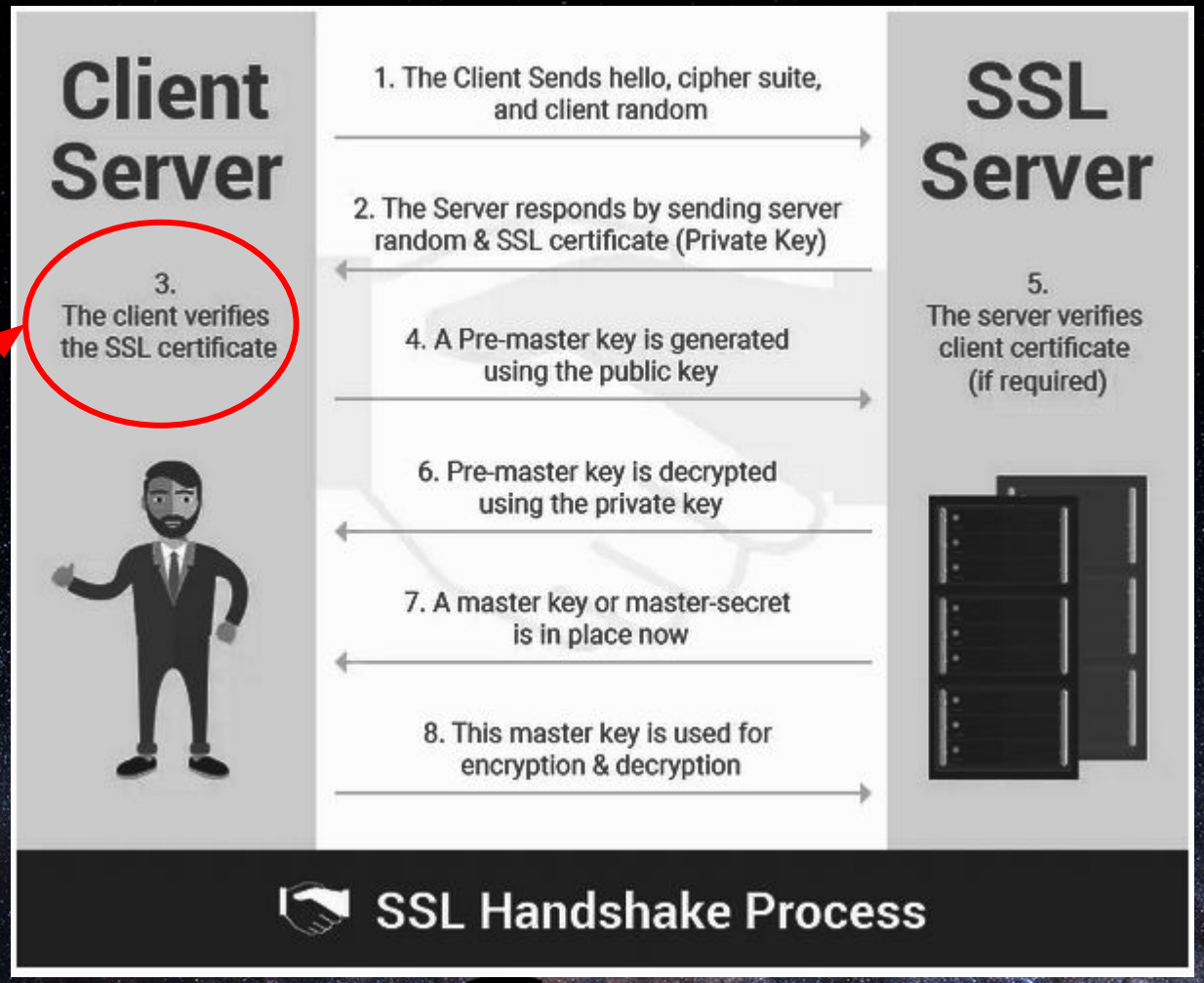
What is SSL

- SSL (Secure Sockets Layer): cryptographic **protocol** used by clients and servers to communicate with each other in order to prevent eavesdropping and tampering
- Deprecated. Successor is Transport Layer Security (TLS)
- Still will see TLS certificates referred to as SSL certificates



What is SSL

Server SSL Certificate



Credit: https://dzone.com/articles/what-is-ssl-how-do-ssl-certificates-work

Why Setup TLS/SSL in Kafka

- Prevent user and password from being sent cleartext over the wire when authenticating
  - Prevents sniffing the wire for passwords (malware)
  - Secures Authorization

- For high volume messaging, SSL has some performance overhead
- To require login to Kafka, you need to implement SASL (Simple Authentication and Security Layer (SASL), RFC 4422, through some challenge and response mechanism:
  - GSSAPI (Kerberos)
  - OAUTHBEARER
  - SCRAM (Salted Challenge Response Authentication Mechanism)
  - **PLAIN**
  - Delegation Tokens
  - LDAP

Why Setup TLS/SSL in Kafka

- Enable Authorization: restrict Kafka topics to certain users through ACLs

How To Setup SSL Certificates

- As of June 2021 the Apache documentation was suspect Version 2.7 (now Version 3.0 looks more thorough)
- Had very little success with it
- Had more success with Confluent's documentation: https://docs.confluent.io/platform/current/kafka/authentication_ssl.html
- Confluent is a commercial company behind Kafka whose founding team created Kafka
- Warning: The documentation includes features not found in Apache Kafka such as Role-based Access Control

How To Setup SSL Certificates


```
-----BEGIN CERTIFICATE-----
MIIEFjCCAv6gAwIBAgIJAJlO1Oh5D3sIMA0GCSqGSIb3DQEBCwUAMIGfMQswCQYD
VQQGEwJVUzEQMA4GA1UECAwHQXJpem9uYTEPMA0GA1UEBwwGVHVjc29uMRAwDgYD
VQQKDAdOT0lSTGFiMRAwDgYDVQQLDAdBTlRBUkVTMSIwIAYDVQpoiBlrYWZrYS5h
bnRakdVzLm5vaXJsYWIuZWR1MSUwIwYJKoZIhvcNAQkBFhZhZGFtLnNjb3R0QG5v
aXJsYWIuZWR1MB4XDTIxMDUyNjIzMDQwNloXDTMxMDUyNDIzMDQwNlowgZ8xCzAJ
BgNVBAYTAlVTMRAwDgYDVQQIDAdBcml6b25hMQ8wDQYDVQQHDAZUdWNzb24xEDAO
BgNVBAoMB05PSVJMYWIxEDAOBgNVBAsMB0FOVEFSRVMxIjAgBgNVBAMMGWthZmth
LmFudGFyZXMubm9PODxhYi5lZHUxJTAjBgkqhkiG9w0BCQEWFmFk98duc2NvdHRA
bm9pcmxhYi5lZHUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDg2wwy
O1mhLi6szm4jIKPNirstdkPDBrjail+l7BQpGzoJ17SWiM27WLLdm53qprte5gno
hF/WR6R+GVnEoNPGUEi04Yapsodifuq93)*DLIJF85jctncmPpldm6ZF4j87Qbm3
GLr7F9333TL3UHwOyEydYDTh8LeJz6+MAczI6iEZqttY6Cc7Ej/wWpyEVwfBf1sH
BL5VcsUwyiqnEpk7rm86InLSj93NFVSmQ7PJ64Wh5YVbBFxOWfsWF7tE1nrfF3k0
2C5OySn3EDE0ququD5K6NN76ji02lkgos9KmBZ+sTYf2fWaDi8ckc8CsfQ3OT+Ki
WelrZn4NvQQmXoKrAgMBAAGjUzBRMB0GA1UdDgQWBBTBTWnj4B0JO6pZmcUyEhBJ
n0SbszAfBgNVHSMEGDAWgBTBTWnj4B0JO6pZmcUyEhBJn0SbszAPBgNVHRMBAf8E
BTADAzmcvMA0GCSqGSIb3DQEBCwUAA4IBAQCJ367+fGwzQxpiN6owrcX6ZNcrlOH
1wd9INg5Vor3byzNOv5oRkas7qp5ynw2cGMPmEIodikB/z373rhWH4/UfbxKUUut
wPwsvFM8YG+clVhWpGIYr2wYEWyMacC+IoD5ZqIUXIh/xzMUaxpsorxMYgUYSoPg
FkkTcIkS8tsbu0mti/vU9wQr6NMEb94Vqq7v5Ka3mPH0Odqlo0ROE3mbX7YKgJ46
sP+4cDoOedoXb6+yx/la0If/vBzGdo8gW4kJSNq8YB+GCqzfIQDT28aOBJUhKRes
z3qIhKjA/K5dmJsBx6uAm5//HRX2QOWIC29KvYUay6nP5t6xKIdo1k70
-----END CERTIFICATE-----
```


- Certificate: file usually in PEM (privacy-enhanced mail) format (2 callout fields, many others)
  - CN (Common Name): name of the object the cert identifies
  - Encryption Key
- Keystore: file that contains a certificate for the Broker's own identity
- Truststore: file that contains all certificate authority certificates that a machine should trust

The Magic: If you have my public key you can encrypt a message to me which can only be decrypted with my private key (keys are really large prime numbers)

ANTARES

How To Setup SSL Certificates

Requirements:

`keytool` installed, comes with a JDK (Java Development Kit), so you will need a JDK installed

`openssl` installed

How To Setup SSL Certificates

Generate broker's certificate with private key

Generate the certificate (containing the private key) and store into the server keystore which will be copied to each broker in the cluster

```
# With user prompts
keytool -keystore kafka.server.keystore.jks -alias localhost -keyalg RSA -genkey
```

This creates a file called a keystore file, named **kafka.server.keystore.jks** (jks for Java Keystore)

Verify:

```
keytool -list -v -keystore kafka.server.keystore.jks
```

The certificate will need to be "vouched for" by signing it with a Certificate Authority

Certificate in a later step

How To Setup SSL Certificates

Create your own Certificate Authority (CA)

This certificate needs to be on the client's truststore and the broker's truststore. (A CA "vouches" for a certificate's authenticity by signing it.)

By creating your own CA, you prevent having to purchase a TLS certificate or requiring another organization to issue a TLS certificate

You have to install it though on your Client. If a malefactor gets this, they can attempt to authenticate, so its value is only to encrypt traffic to prevent sniffing the wire.

Create ca-key and ca-cert.pm:

```
openssl req -new -x509 -keyout ca-key -out ca-cert.pem -days {validity}
```

Let's Encrypt is an open source Certificate Authority: Its CA is on your computer that your browser knows how to lookup. There are many installed on you computer already. https://ui.adsabs.harvard.edu/ cert is verified by Internet2 for example.

How To Setup SSL Certificates

Create your own Certificate Authority (CA)

Add the CA file to your clients:

For kafka-python, the way to use the CA file:

```
consumer = KafkaConsumer('my-topic',
                         group_id='my-group',
                         bootstrap_servers=['localhost:9092'],
                         ssl_cafile='ca-cert.pem')
```

How To Setup SSL Certificates

Create your own Certificate Authority (CA)

Add the CA file to your broker's truststore (so it will trust the CA)

```
keytool -keystore kafka.server.truststore.jks -alias CARoot -importcert -file ca-cert.pem
```

Copy kafka.server.truststore.jks to your server if it's not already there. Reference it in the server config: `/opt/kafka/config/server.properties`

```
ssl.truststore.location=/opt/kafka/config/cert3rdparty/kafka.server.truststore.jks
ssl.truststore.password= <secretpassword>
```

How To Setup SSL Certificates

Sign the certificate with the CA:

Now we have a certificate that we can use to encrypt the connection and a custom Certificate Authority certificate to "vouch" for the certificate. Recall in slide 10 we created the certificate and put it in kafka.server.keystore.jks

Create a certificate signing request (CSR) from the keystore to a standalone file:

```
keytool -keystore kafka.server.keystore.jks -alias localhost -certreq -file cert-file
```

Using the ca-cert.pem, ca-key, and CSR files, sign the cert

```
openssl x509 -req -CA ca-cert.pem -CAkey ca-key -in cert-file -out cert-signed -days {validity} -CAcreateserial -passin pass:{ca-password}
```

This makes the CA "vouch" for the certificate.

How To Setup SSL Certificates

Import the the certificate of the CA and the signed certificate into broker keystore

```
keytool -keystore kafka.server.keystore.jks -alias CARoot -importcert -file ca-cert.pem

keytool -keystore kafka.server.keystore.jks -alias localhost -importcert -file \
 cert-signed
```

How To Setup SSL Certificates

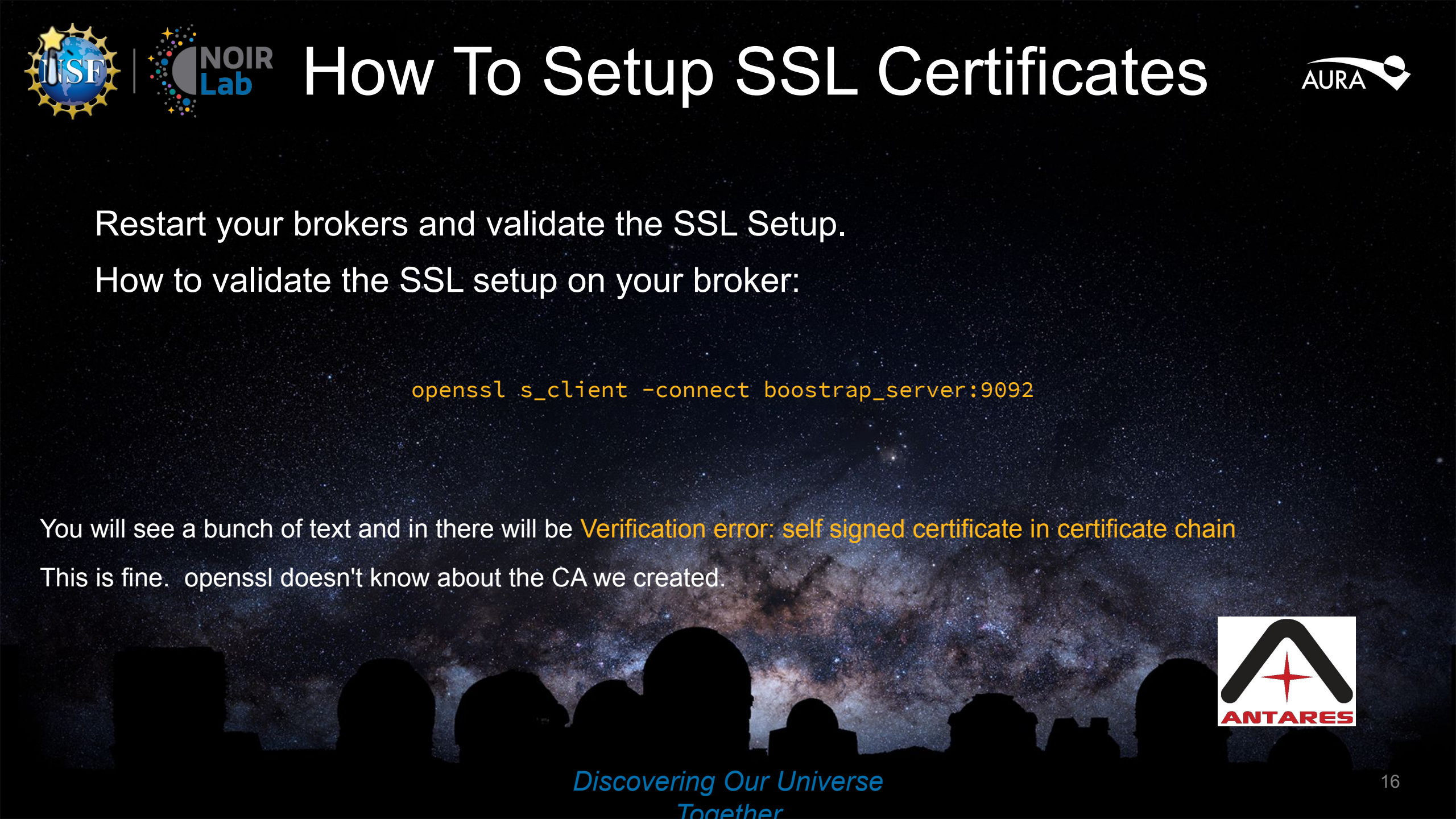
Restart your brokers and validate the SSL Setup.

How to validate the SSL setup on your broker:

```
openssl s_client -connect boostrap_server:9092
```

You will see a bunch of text and in there will be Verification error: self signed certificate in certificate chain

This is fine. openssl doesn't know about the CA we created.



You're done!

Next you will want to implement Authentication, SASL/PLAIN is the easiest.

To lock-down topics, you will then want to implement Authorization