Business Domain Modelling

raj@acloudfan.com

@acloudfan

http://ACloudFan.com

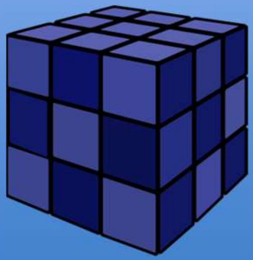
Discount Coupon Links to UDEMY courses:

https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099

mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com



This deck is part of a online course on "Hyperledger Fabric Development with Composer"

Transactions & Events

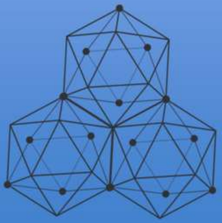
raj@acloudfan.com

@acloudfan

http://ACloudFan.com

Learning Objectives:

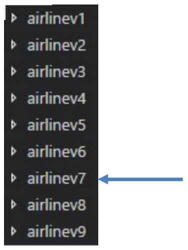
- Transactions
- Events



PS: High level overview of the code; details in next section

Incremental Creation of **ACME Air Domain Model**

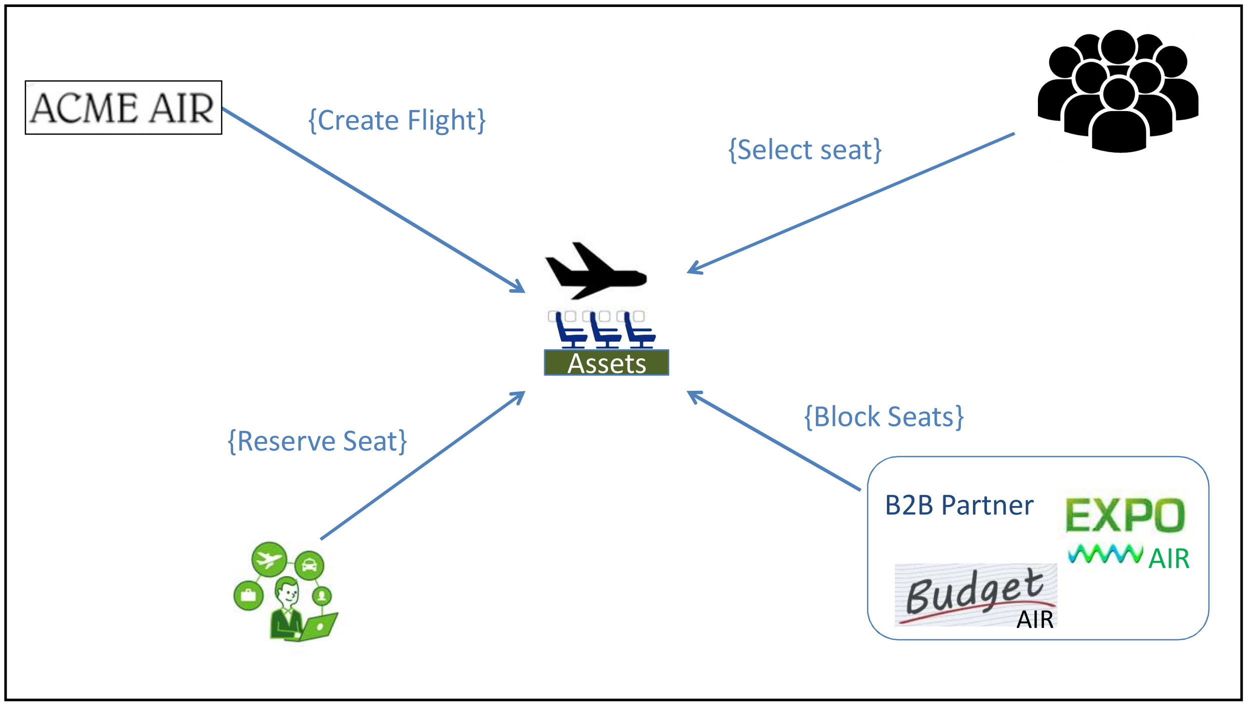
https://github.com/acloudfan/**HLF-Course-Domain-Model**

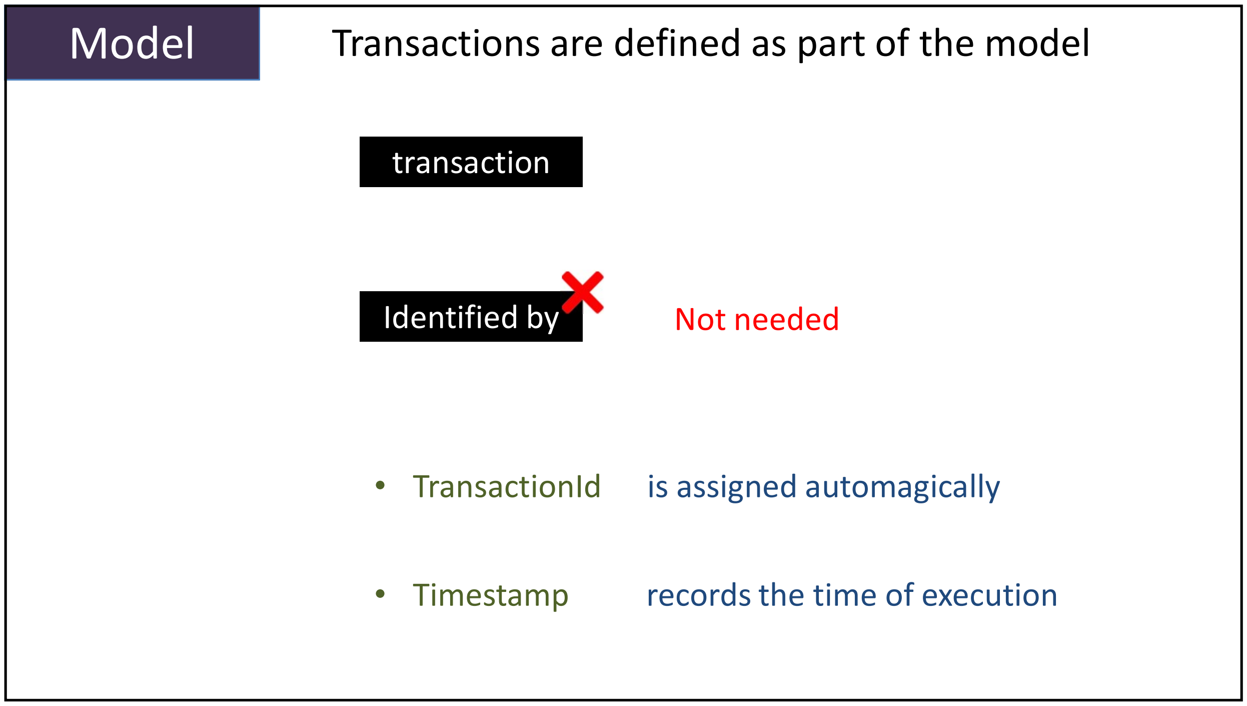

airlinev1
airlinev2
airlinev3
airlinev4
airlinev5
airlinev6
airlinev7
airlinev8
airlinev9

Transactions

Actions that participants can take on assets

- State changes of an asset i.e., world state
- All transactions recorded in the ledger
  - State may be recreated by replaying the transaction





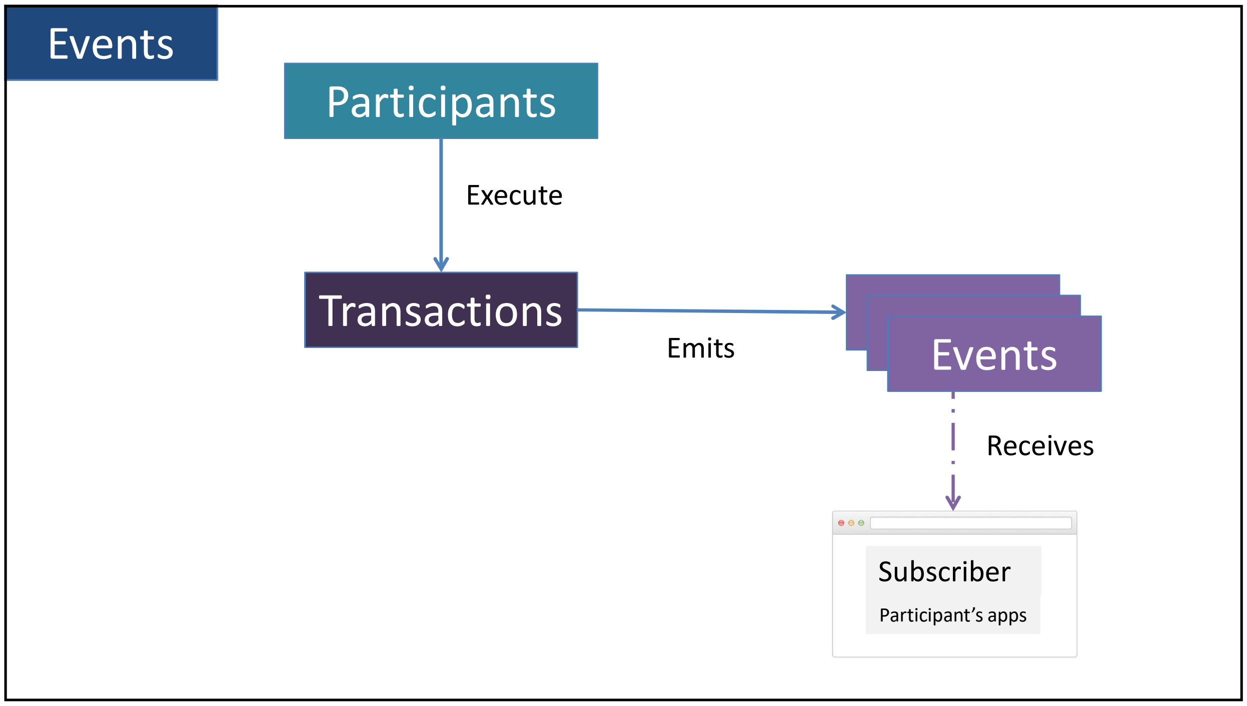
Model

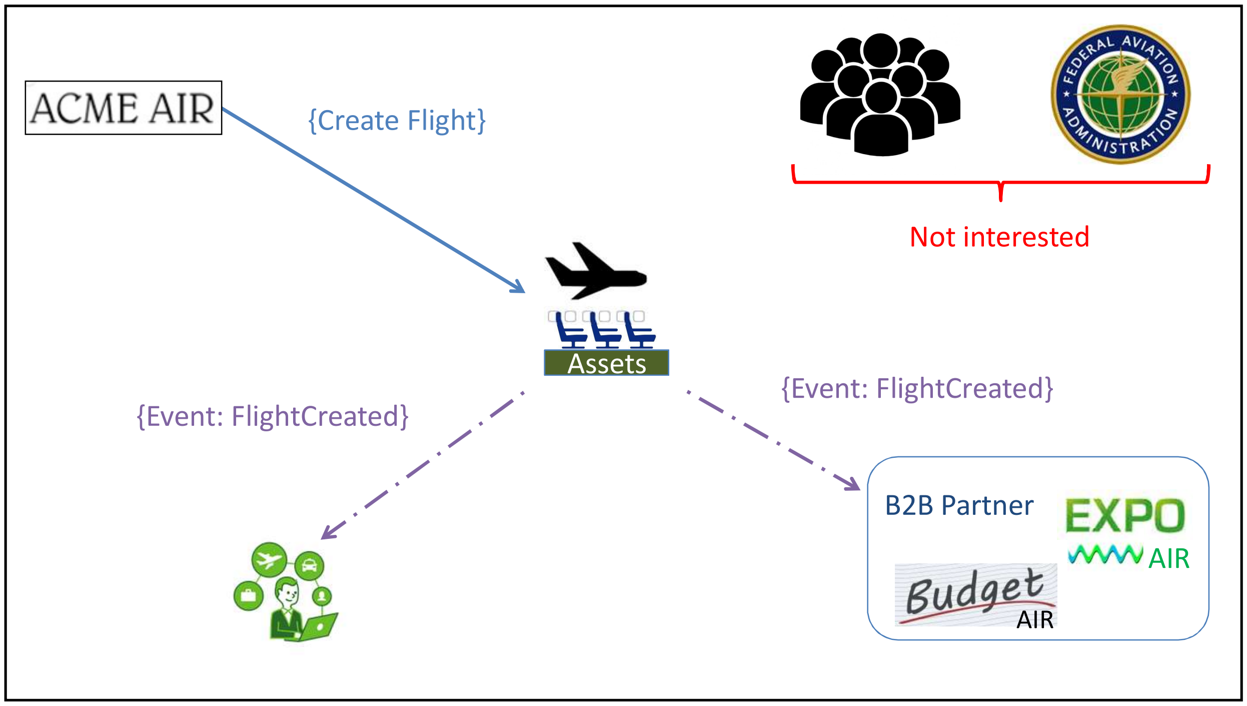
Transactions are defined as part of the model

transaction

~~Identified by~~ Not needed

- TransactionId is assigned automagically
- Timestamp records the time of execution


Events
Participants
Execute
Transactions
Emits
Events
Receives
Subscriber
Participant's apps


ACME AIR
{Create Flight}
Not interested
Assets
{Event: FlightCreated}
{Event: FlightCreated}
B2B Partner
EXPO AIR
Budget AIR

Model

Events are defined as part of the model

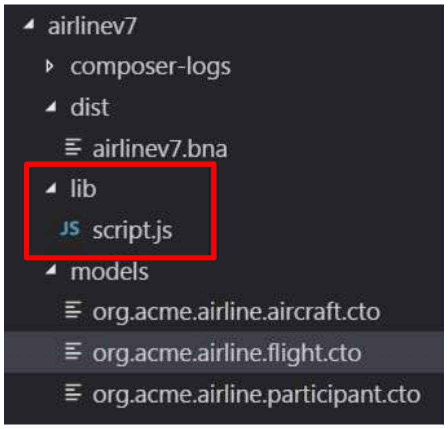
event

Identified by ✗ Not needed

- eventId is assigned automagically
- timestamp records the time of execution

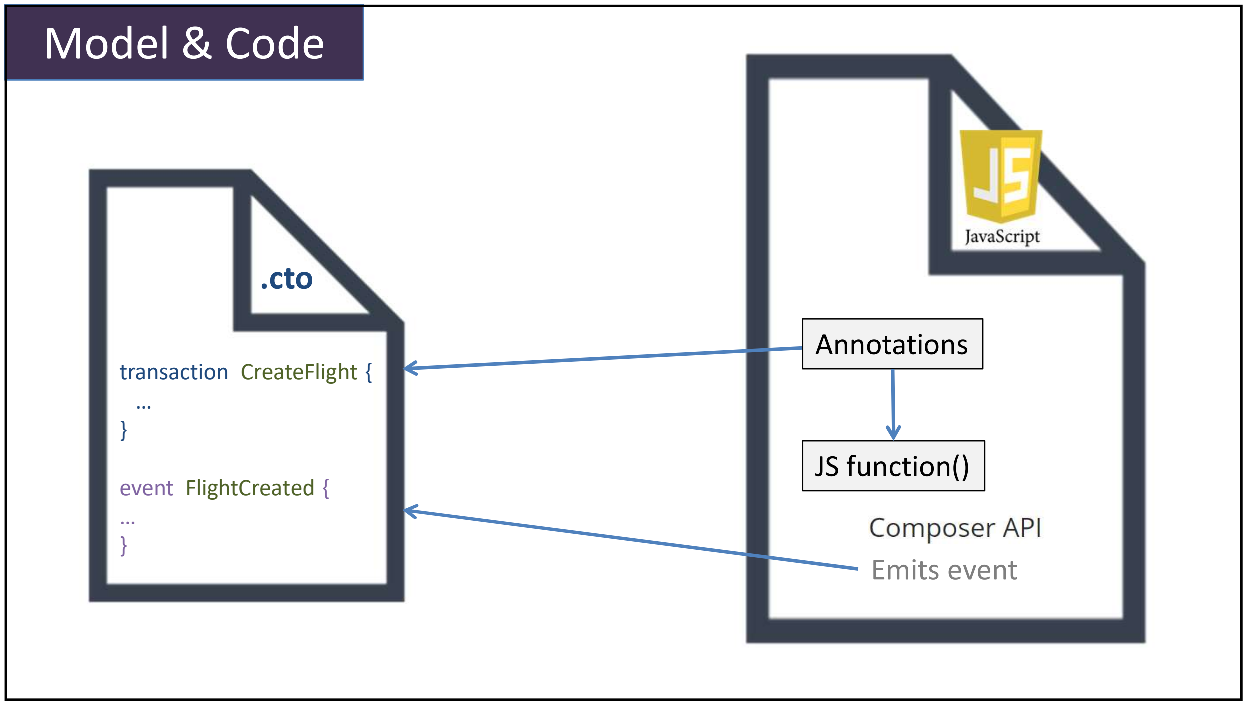
Transaction Logic

- Transactions are coded in JavaScript

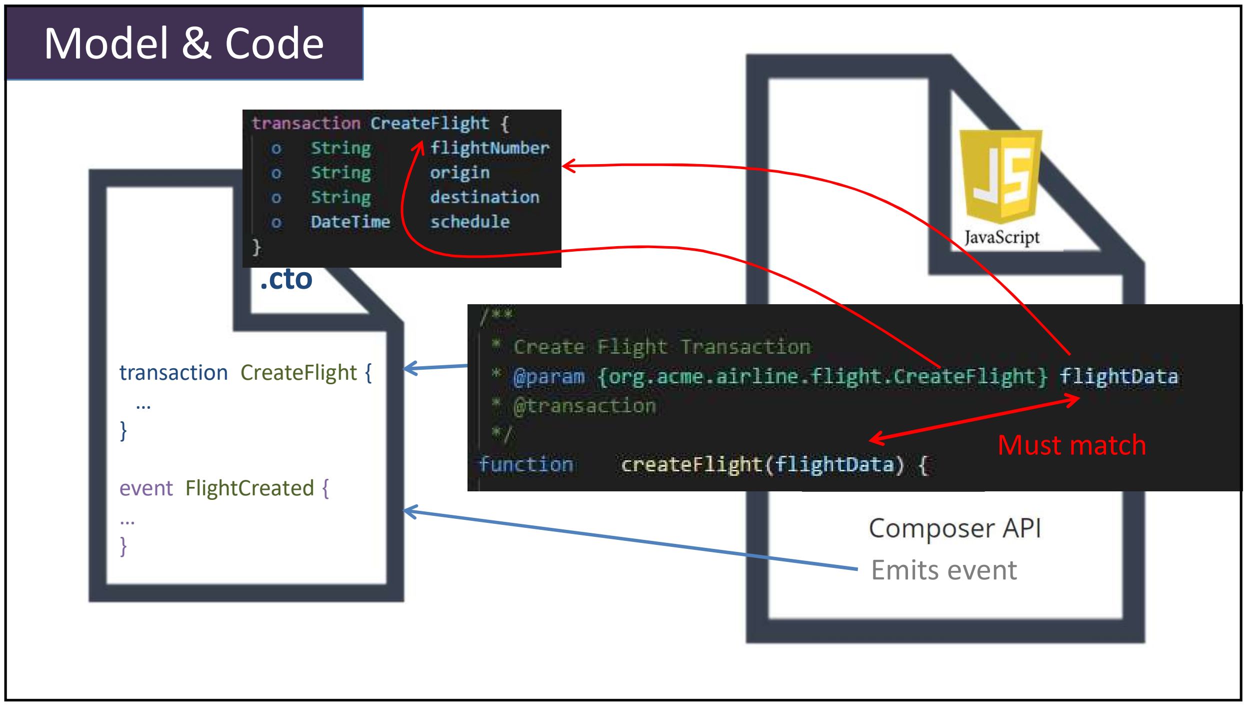

airlinev7
composer-logs
dist
airlinev7.bna
lib
script.js
models
org.acme.airline.aircraft.cto
org.acme.airline.flight.cto
org.acme.airline.participant.cto

- Logic may be spread across multiple JS files
- Annotations connects the code to the model

Model & Code



Model & Code



```
transaction CreateFlight {
  o   String     flightNumber
  o   String     origin
  o   String     destination
  o   DateTime   schedule
}
```

```
/**
 * Create Flight Transaction
 * @param {org.acme.airline.flight.CreateFlight} flightData
 * @transaction
 */
function    createFlight(flightData) {
```

Historian

- Registry to record of all successful transactions
  - System defined asset [HistorianRecord] to track
  - Tracks system transactions as well
  - Historian records may be queried

Query Language

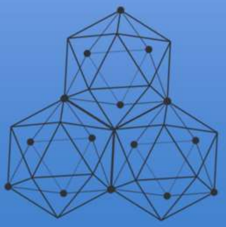
raj@acloudfan.com

@acloudfan

http://ACloudFan.com

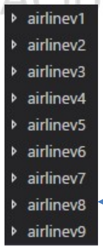
Learning Objectives:

- Named queries
- Walkthrough of query(s)



Incremental Creation of **ACME Air Domain Model**

https://github.com/acloudfan/**HLF-Course-Domain-Model**

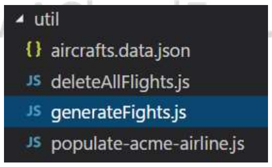


```
airlinev1
airlinev2
airlinev3
airlinev4
airlinev5
airlinev6
airlinev7
airlinev8
airlinev9
```

Code shown in video may change over time

Hyperledger Fabric API

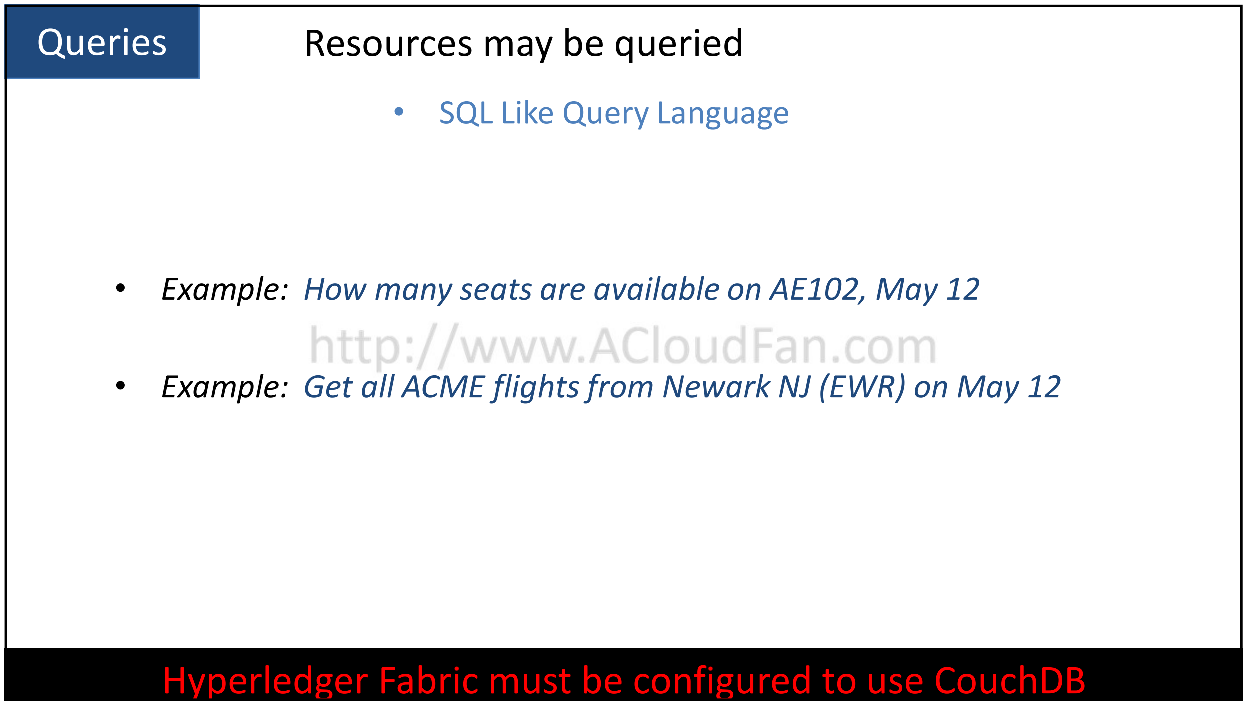
https://github.com/acloudfan/**HLF-Fabric-API**



```
util
  aircrafts.data.json
  deleteAllFlights.js
  generateFights.js
  populate-acme-airline.js
```

<<< Generates Test Data

Code shown in video may change over time



Queries

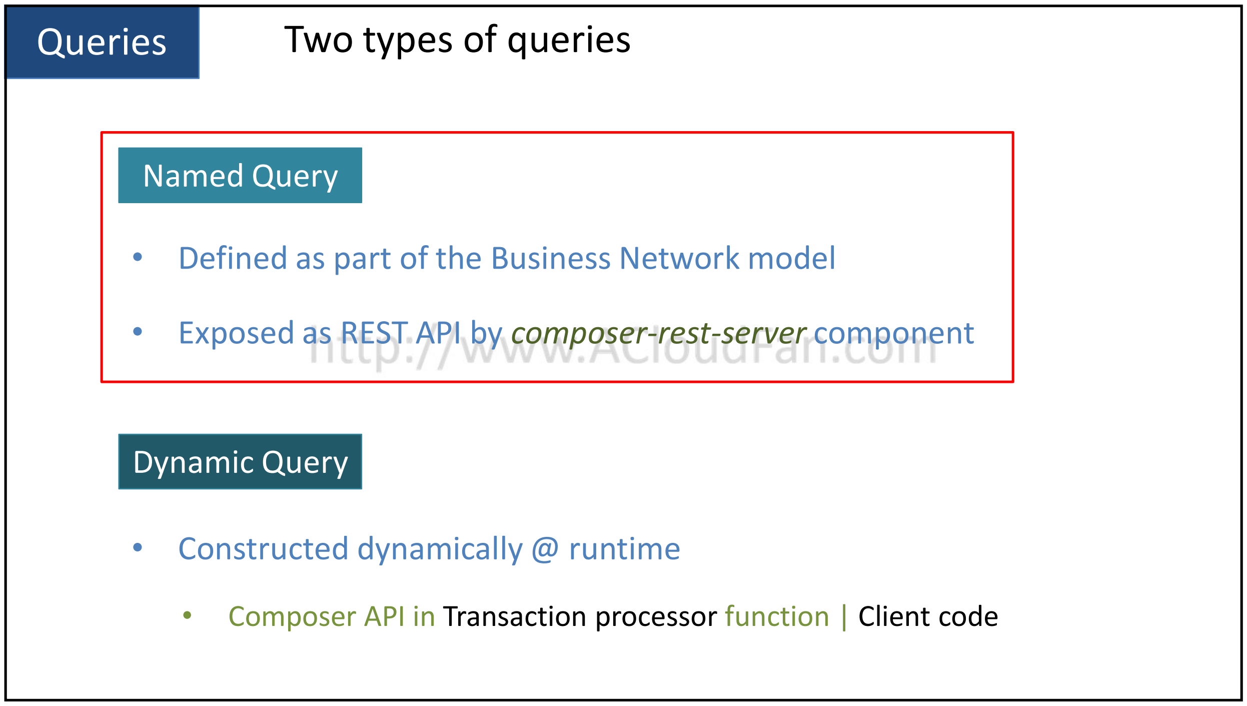
Resources may be queried

- SQL Like Query Language

- *Example: How many seats are available on AE102, May 12*

- *Example: Get all ACME flights from Newark NJ (EWR) on May 12*

Hyperledger Fabric must be configured to use CouchDB



Queries

Two types of queries

Named Query

- Defined as part of the Business Network model
- Exposed as REST API by *composer-rest-server* component

Dynamic Query

- Constructed dynamically @ runtime
  - Composer API in Transaction processor function | Client code

Named Query

- All queries defined in a single file called queries.qry

query {

description: *Provide the description of the query*

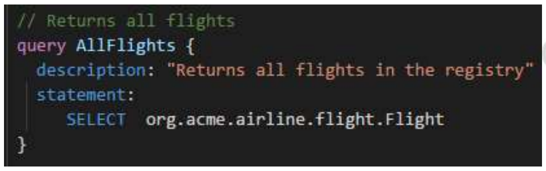
statement: *Using composer query language*

}

Query Language

SELECT

- Mandatory operator
- Defines the Registry & Asset or Participant type



```
// Returns all flights
query AllFlights {
  description: "Returns all flights in the registry"
  statement:
      SELECT  org.acme.airline.flight.Flight
}
```

FROM

- Optional operator
- Defines a different registry to query

Query Language

Queries allow use of parameters

- Uses the _${param-name} syntax

  _$flightNumber _$origin_airport

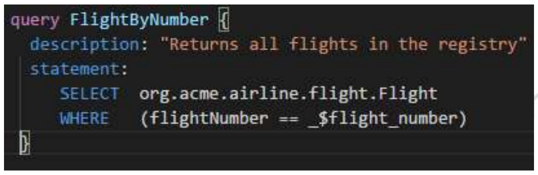
- Only primitive types allowed

  - String
  - Integer
  - Double
  - Long
  - Boolean
  - DateTime

Query Language

WHERE

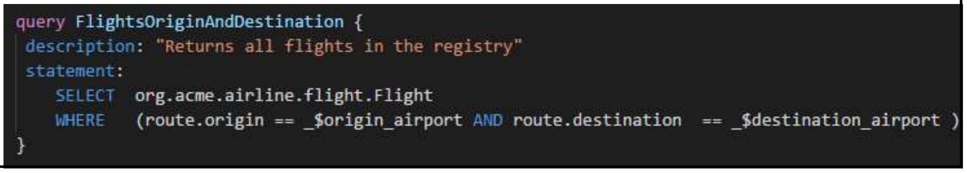
- Optional operator
- Defines the conditions to be applied to the registry data



```
query FlightByNumber {
  description: "Returns all flights in the registry"
  statement:
     SELECT  org.acme.airline.flight.Flight
     WHERE   (flightNumber == _$flight_number)
}
```

AND OR

- Logical operators

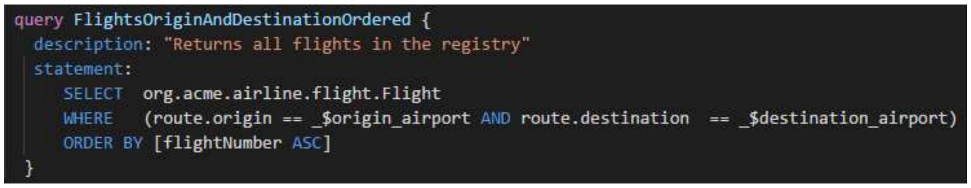


```
query FlightsOriginAndDestination {
 description: "Returns all flights in the registry"
 statement:
    SELECT  org.acme.airline.flight.Flight
    WHERE   (route.origin == _$origin_airport AND route.destination  == _$destination_airport )
}
```

Query Language

ORDER BY

- Optional operator
- Defines the sorting of results
- ASC | DESC



```
query FlightsOriginAndDestinationOrdered {
  description: "Returns all flights in the registry"
  statement:
     SELECT  org.acme.airline.flight.Flight
     WHERE   (route.origin == _$origin_airport AND route.destination  == _$destination_airport)
     ORDER BY [flightNumber ASC]
  }
```

CONTAINS

- Optional operator
- Applies to array attribute

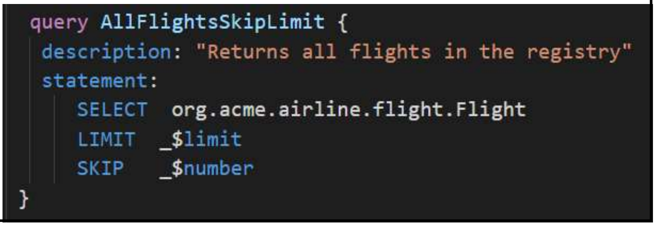
Query Language

LIMIT

- Optional operator
- Defines the maximum number of results to return from a query
- Default limit = 25

SKIP

- Optional operator
- Defines the number of results to skip



```
query AllFlightsSkipLimit {
 description: "Returns all flights in the registry"
 statement:
    SELECT  org.acme.airline.flight.Flight
    LIMIT  _$limit
    SKIP   _$number
}
```

Identity Management

raj@acloudfan.com

@acloudfan

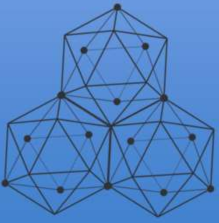
http://ACloudFan.com

Learning Objectives:

- Relationship between Participant & Identity

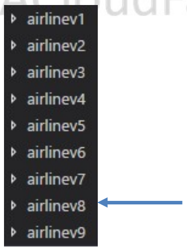
Playground

CLI Tool



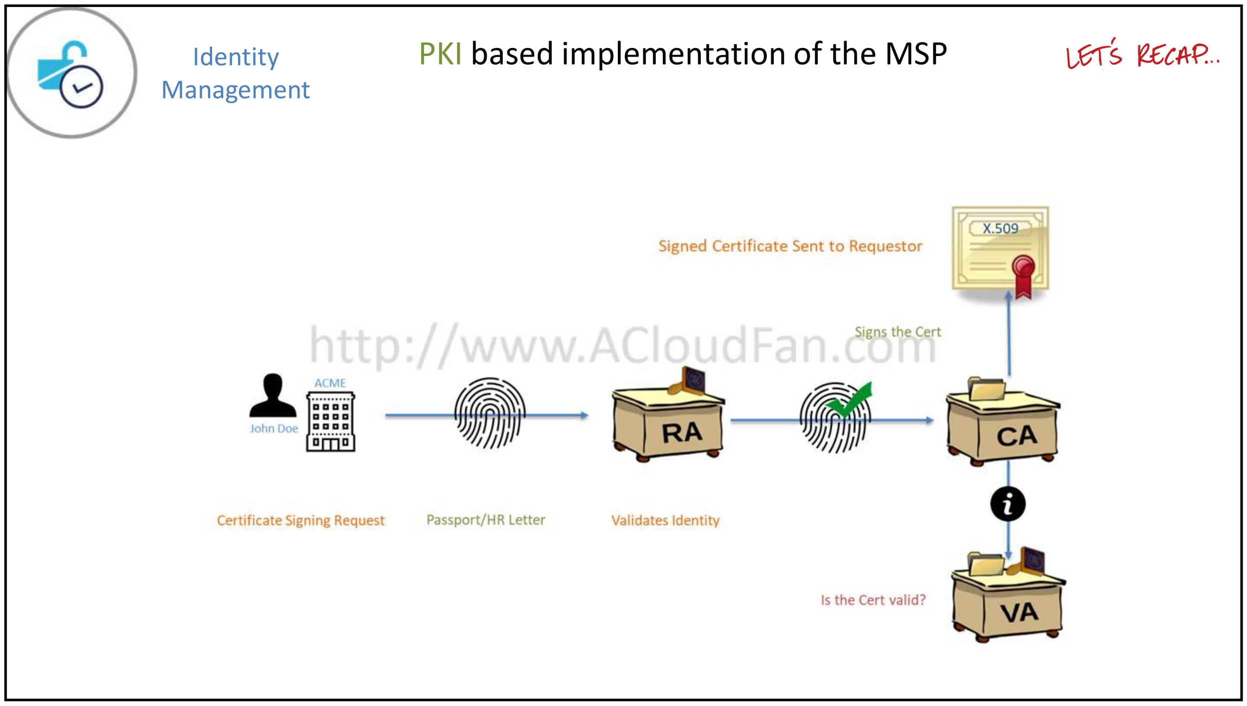
Incremental Creation of **ACME Air Domain Model**

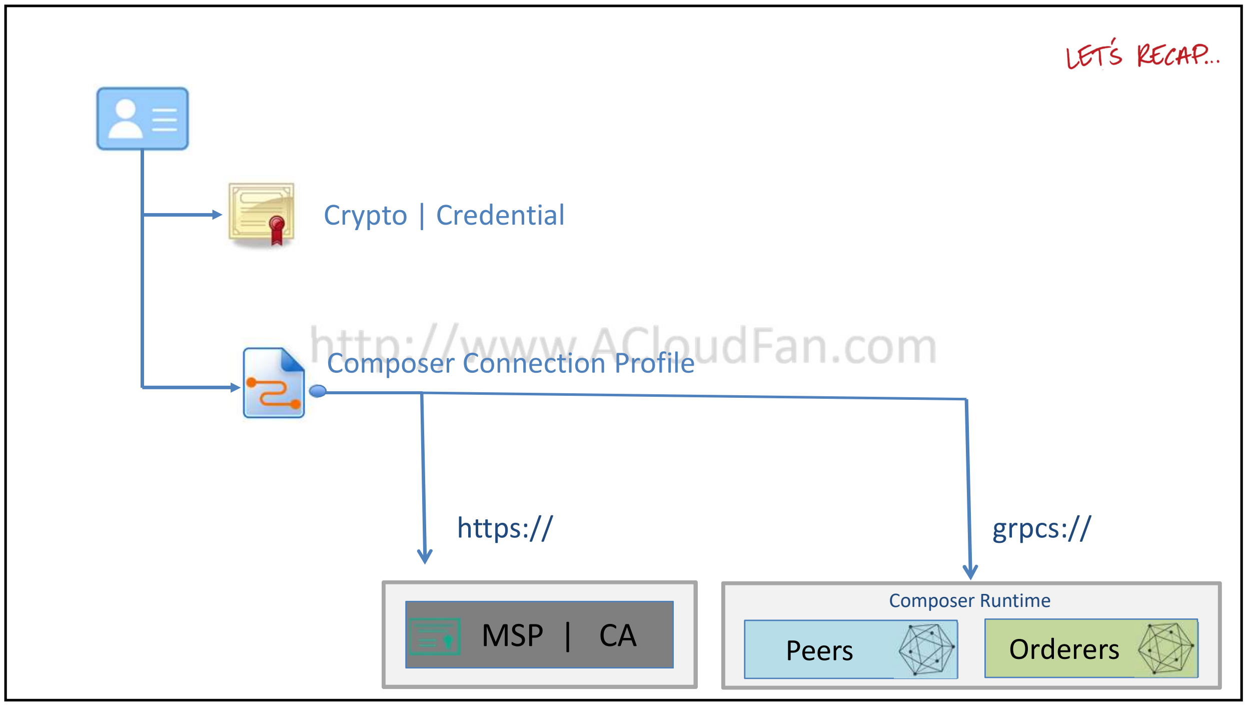
https://github.com/acloudfan/**HLF-Course-Domain-Model**

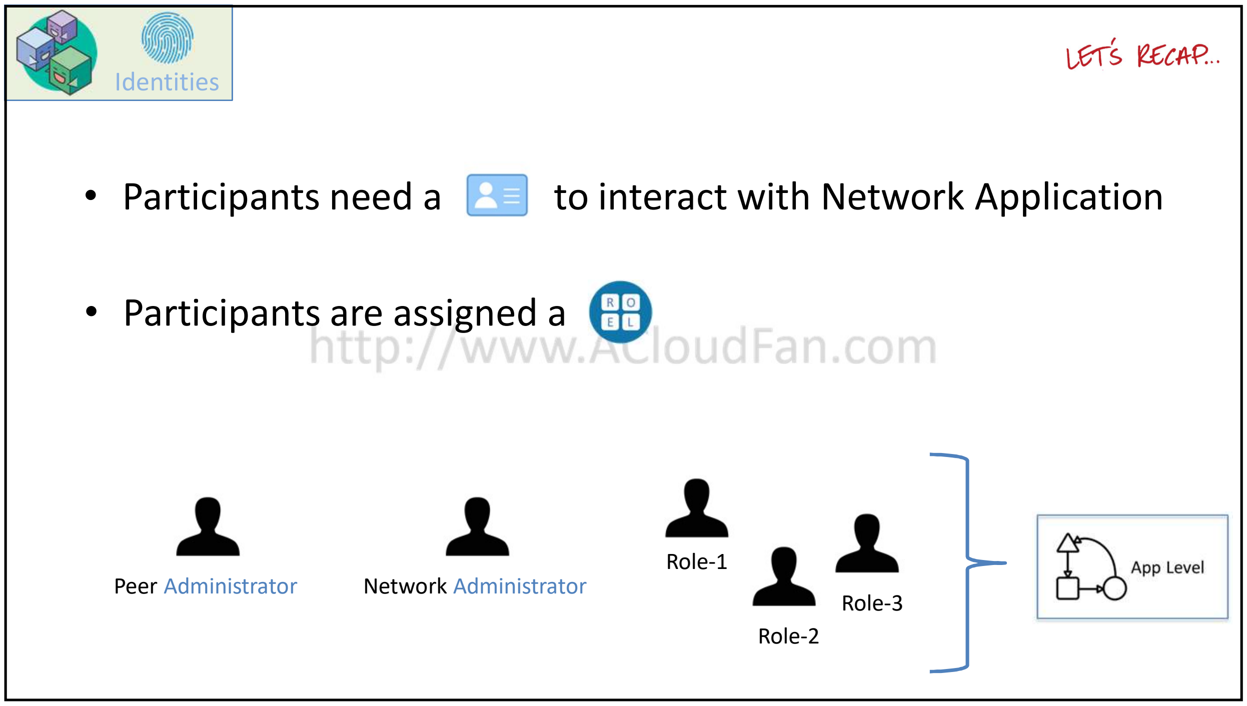


```
airlinev1
airlinev2
airlinev3
airlinev4
airlinev5
airlinev6
airlinev7
airlinev8
airlinev9
```

Code shown in video may change over time


Identity Management
PKI based implementation of the MSP
LET'S RECAP...
Signed Certificate Sent to Requestor
X.509
Signs the Cert
ACME
John Doe
RA
CA
Certificate Signing Request
Passport/HR Letter
Validates Identity
Is the Cert valid?
VA


LET'S RECAP...
Crypto | Credential
Composer Connection Profile
https://
grpcs://
MSP | CA
Composer Runtime
Peers
Orderers

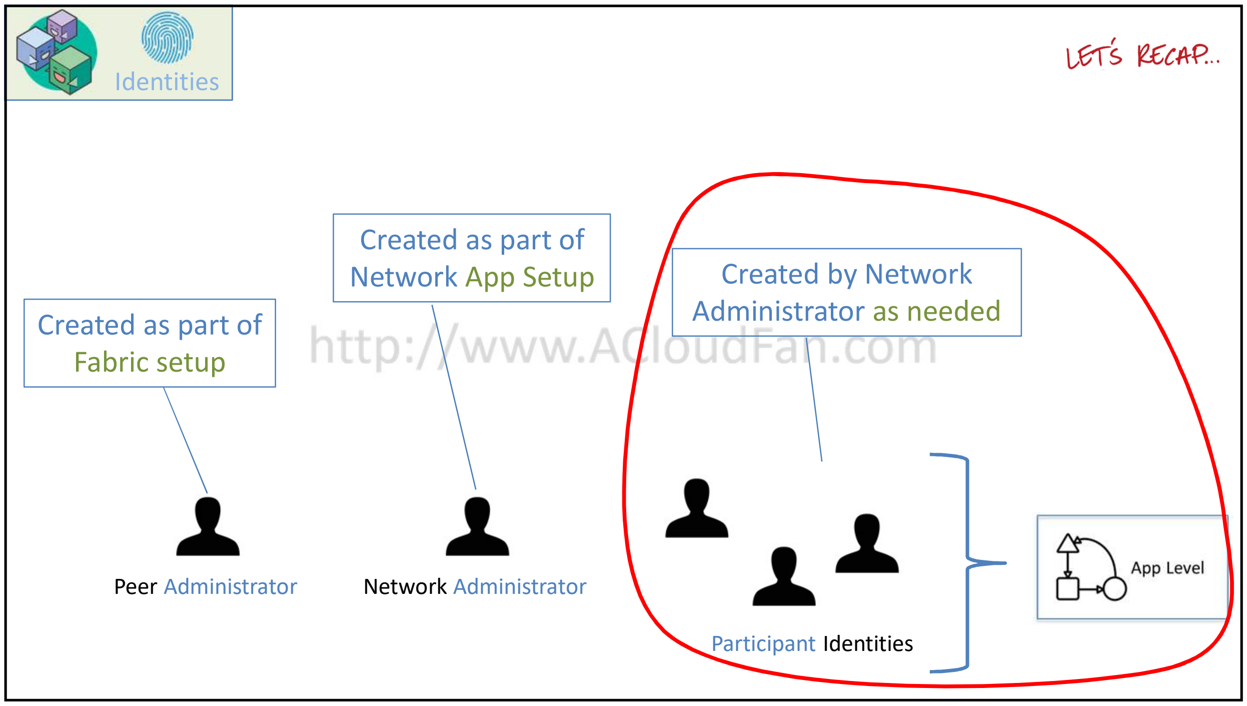


Identities

LET'S RECAP...

- Participants need a to interact with Network Application
- Participants are assigned a

Peer Administrator
Network Administrator
Role-1
Role-2
Role-3
App Level



Identities

LET'S RECAP...

Created as part of Fabric setup
Created as part of Network App Setup
Created by Network Administrator as needed
Peer Administrator
Network Administrator
Participant Identities
App Level

CLI Tool

User of tool MUST have authority to manage identities

Provide appropriate card with –c option

```
composer  participant  add   --help
```

```
composer  identity --help
```

CLI Tool

User of tool MUST have authority to manage identities

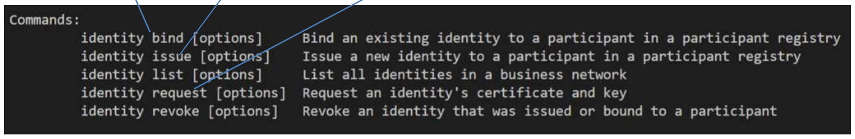
Provide appropriate card with –c option

Binds existing Identity (cert) to Participant

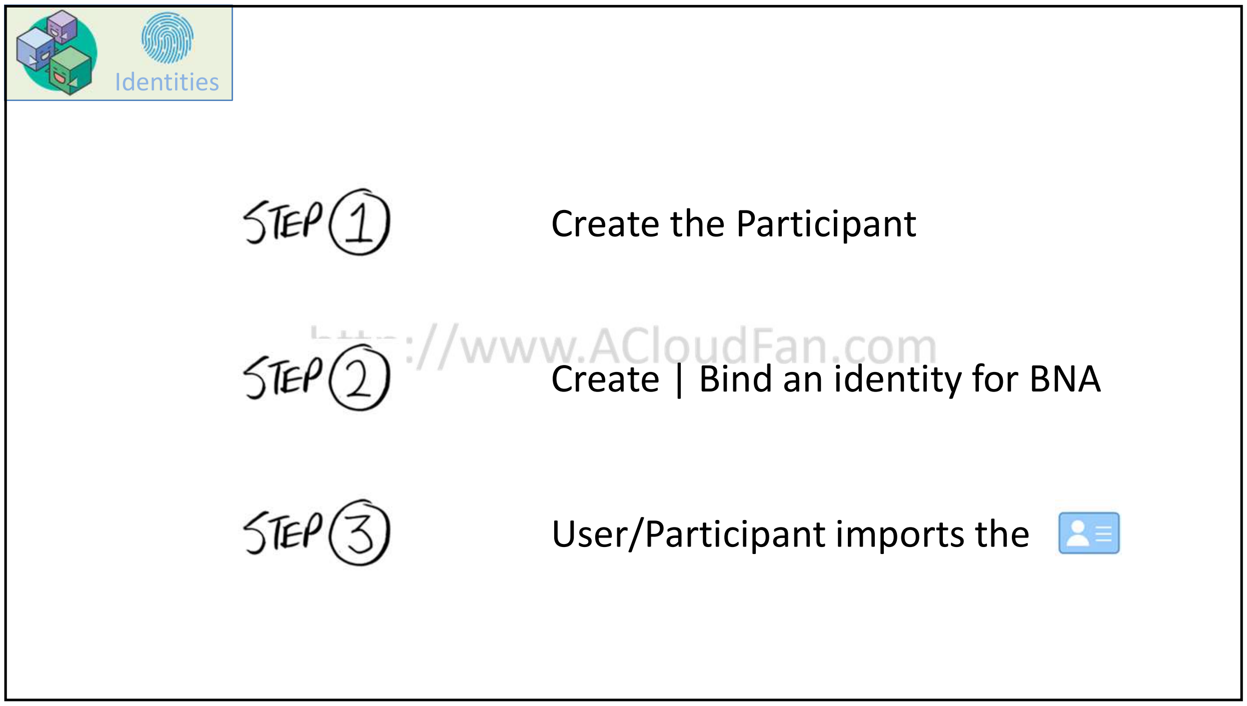
Issuing new Identity (card) to Participant

Get the For the identity

```
composer  identity --help
```



```
Commands:
         identity bind [options]     Bind an existing identity to a participant in a participant registry
         identity issue [options]    Issue a new identity to a participant in a participant registry
         identity list [options]     List all identities in a business network
         identity request [options]  Request an identity's certificate and key
         identity revoke [options]   Revoke an identity that was issued or bound to a participant
```

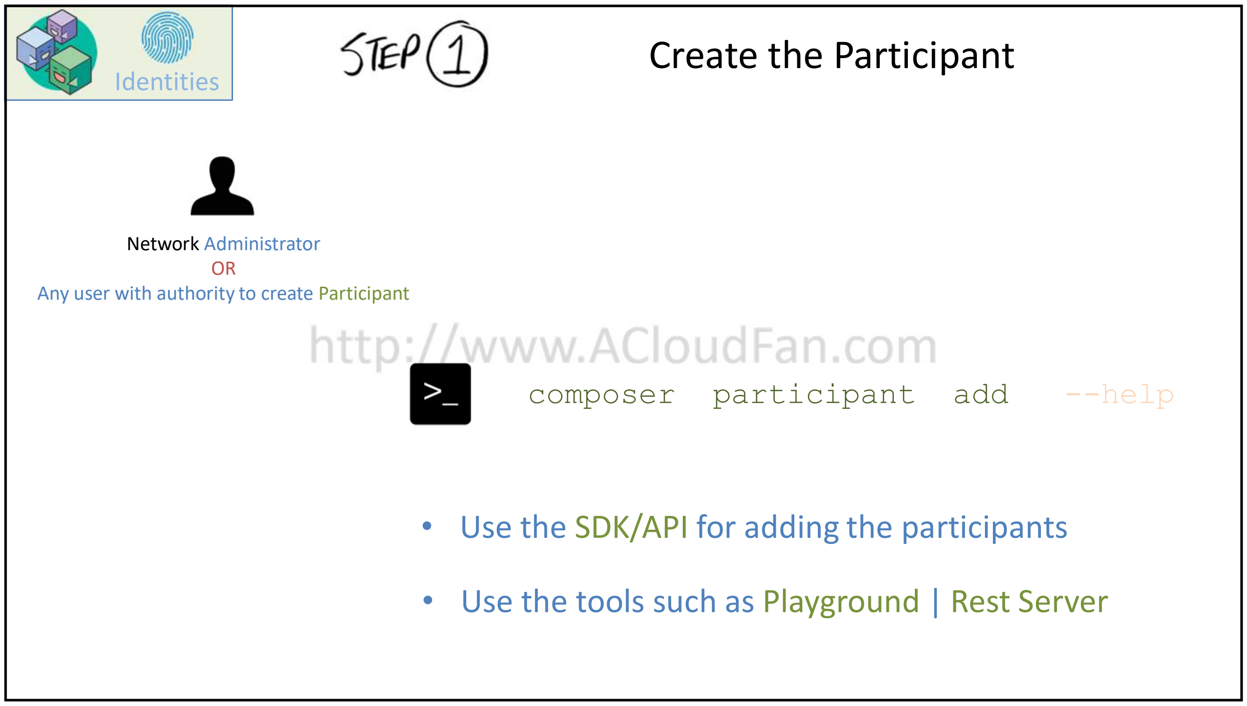


Identities

STEP 1 Create the Participant

STEP 2 Create | Bind an identity for BNA

STEP 3 User/Participant imports the



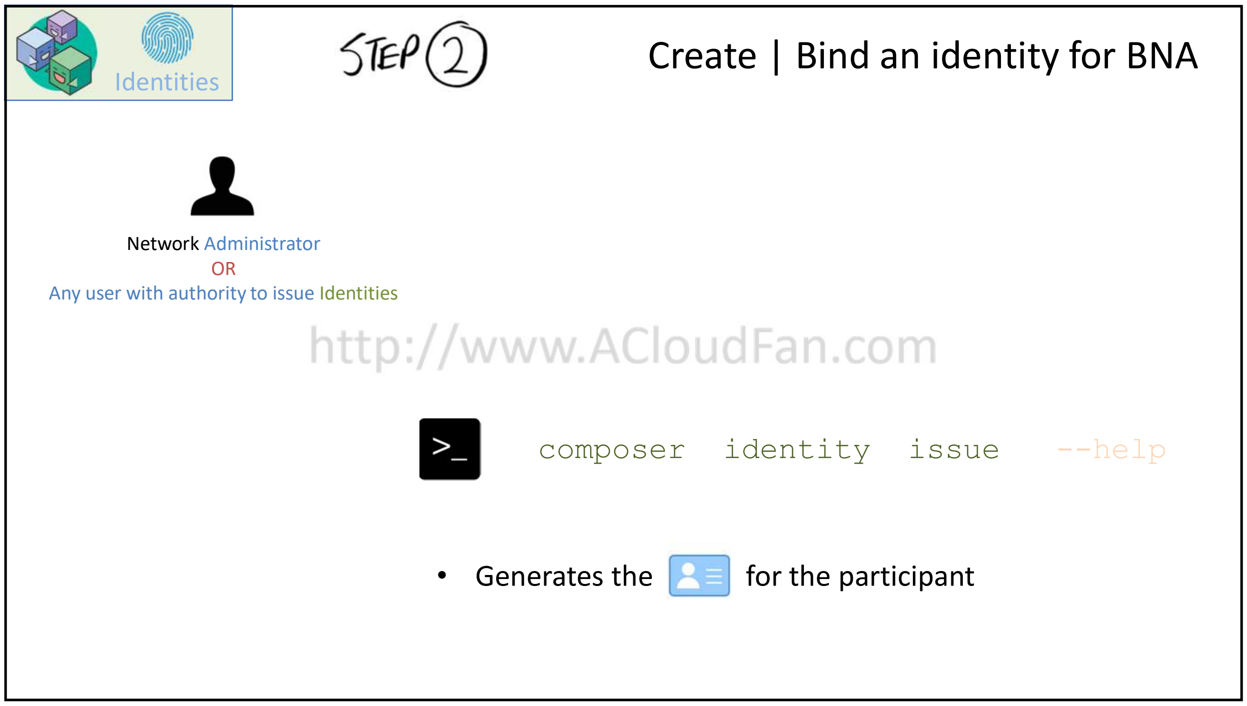
Identities

STEP 1 Create the Participant

Network Administrator
OR
Any user with authority to create Participant

```
>_   composer  participant  add   --help
```

- Use the SDK/API for adding the participants
- Use the tools such as Playground | Rest Server



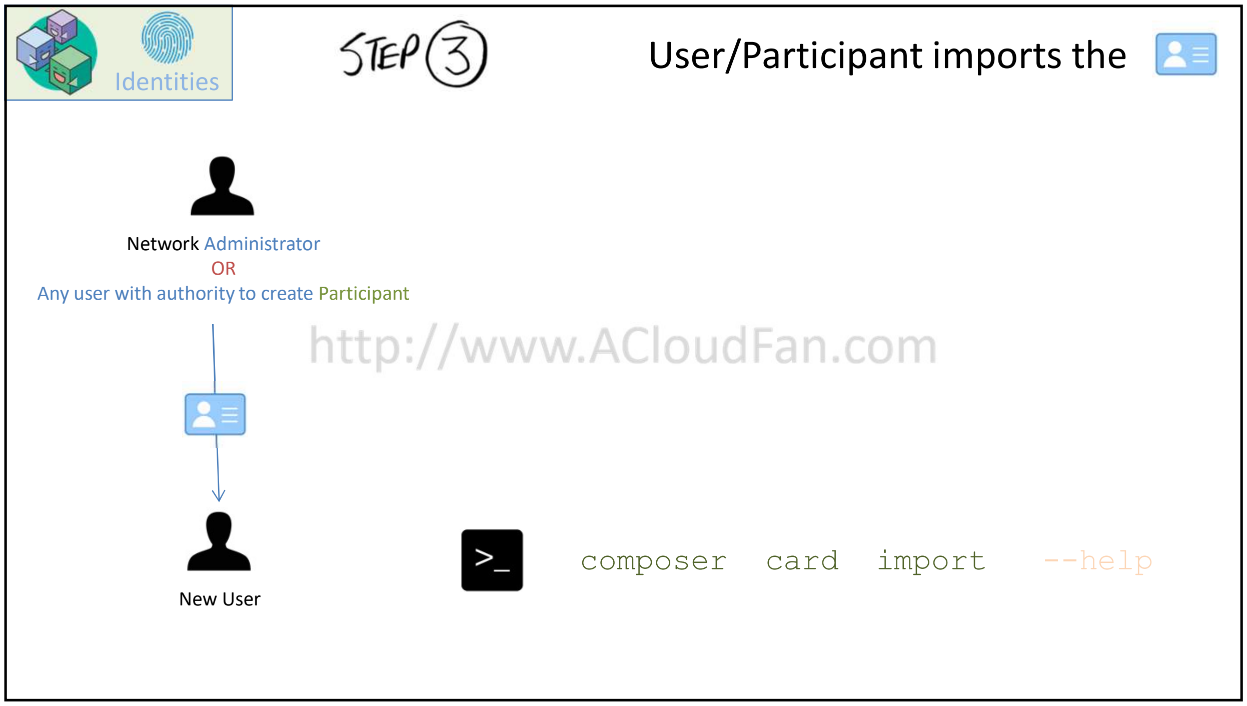
STEP 2 Create | Bind an identity for BNA

Identities

Network Administrator
OR
Any user with authority to issue Identities

```
>_   composer identity issue --help
```

- Generates the for the participant



STEP 3 User/Participant imports the

Identities

Network Administrator
OR
Any user with authority to create Participant

New User

```
>_   composer card import --help
```

Identities

Walkthrough #1 Playground

Walkthrough #2 CLI Tool

Identities

STEP 1 Create the Participant

```
>_ composer participant add -d {
      "$class":"org.acme.airline.participant.ACMENetworkAdmin",
      "participantKey":"johnd",
      "contact":{
         "$class":"org.acme.airline.participant.Contact",
         "fName":"John",
         "lname":"Doe",
         "email":"john.doe@acmeairline.com"
      }
   }
   -c cardName
```

Identities

STEP 2

Create | Bind an identity for BNA

```
composer identity issue  -u johnd
                         -a org.acme.airline.participant.ACMENetworkAdmin#johnd
                         -c cardName
                         -x
```

Controls if the user can manage identities >> `-x`

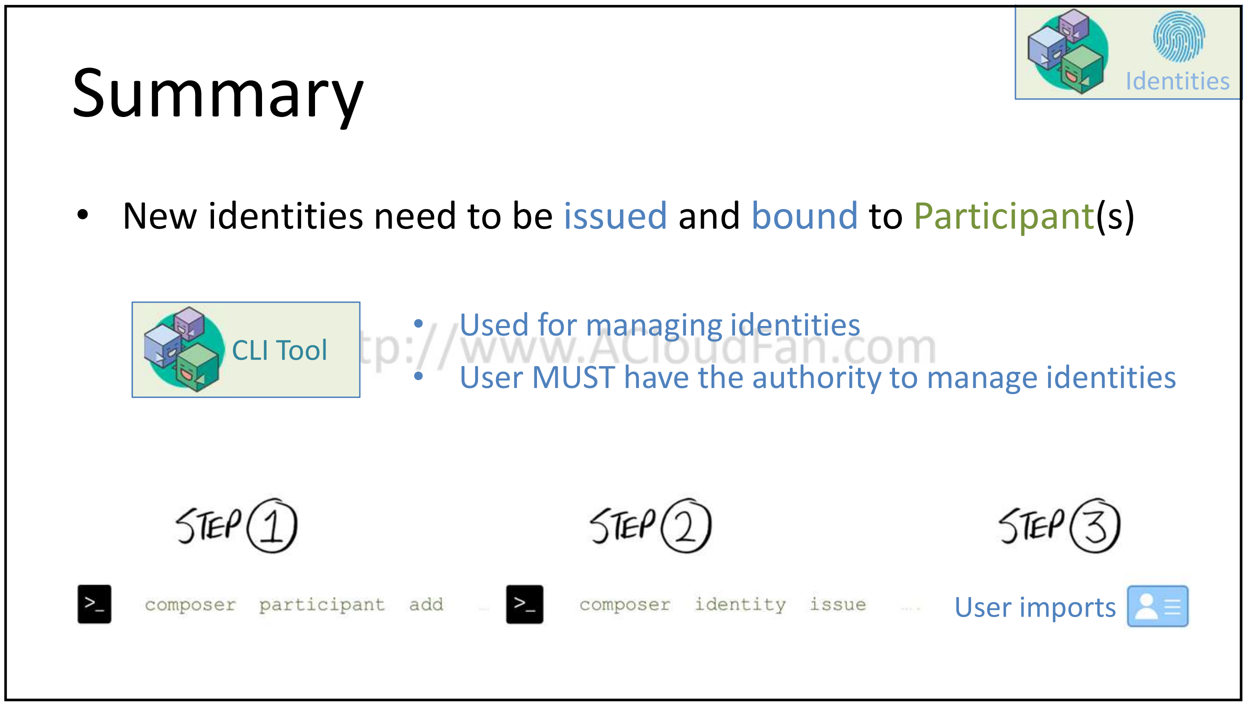
Identities

STEP 3

User/Participant imports the 🪪

New User

```
composer card import  -f johnd@airlinev8.card
```



Summary

Identities

- New identities need to be issued and bound to Participant(s)

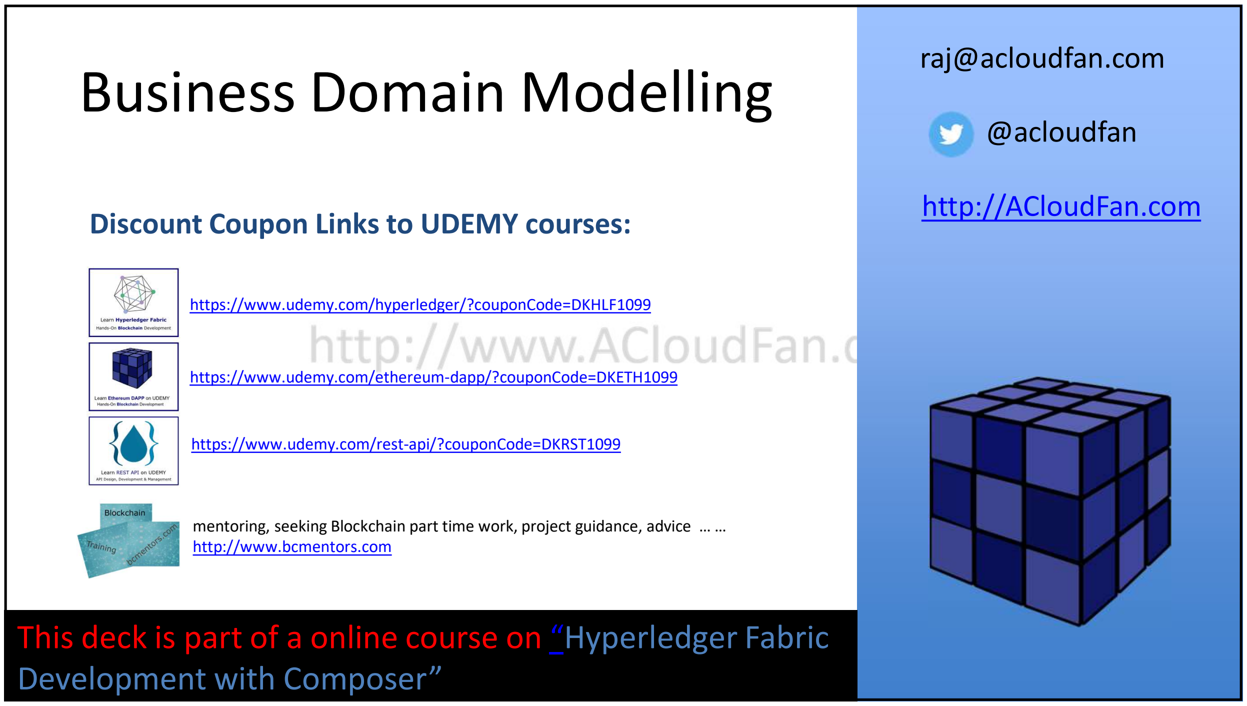
CLI Tool

- Used for managing identities
- User MUST have the authority to manage identities

STEP 1
STEP 2
STEP 3

```
>_ composer participant add ...
>_ composer identity issue ....
```

User imports



Business Domain Modelling

raj@acloudfan.com

@acloudfan

http://ACloudFan.com

Discount Coupon Links to UDEMY courses:

https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099

mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com

This deck is part of a online course on "Hyperledger Fabric Development with Composer"

Access Control Language

Part 1 of 2 : Simple Rule

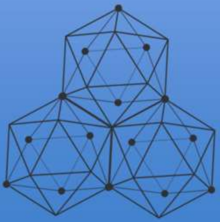
raj@acloudfan.com

@acloudfan

http://ACloudFan.com

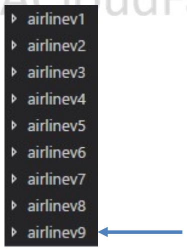
Learning Objectives:

- Access Control Language
- Rule processing @ runtime
- Walkthrough of a simple rule



Incremental Creation of **ACME Air Domain Model**

https://github.com/acloudfan/**HLF-Course-Domain-Model**



```
airlinev1
airlinev2
airlinev3
airlinev4
airlinev5
airlinev6
airlinev7
airlinev8
airlinev9
```

Code shown in video may change over time

Network Access

?

Should the participants be able to access all resources & take any action on Network Application

Access Control

Access Control Implementation

Programmatic

Coded in the transaction processing functions

- Based on the user context & transaction data

Declarative ✓

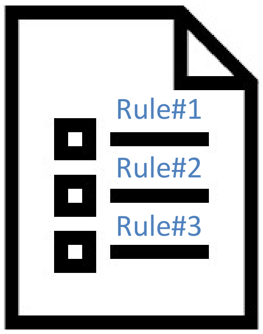
Rules defined *Access Control Language*

Access Control Language

“ Access Control Language provides Declarative access control over the elements of the domain model

Access Control Language

Single file contain all Rules

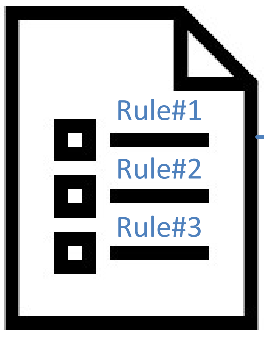


permissions.acl

Everything permitted if this file is missing

Access Control Language

Single file contain all Rules permissions.acl



```
rule AllAccess {
  description: "AllAccess - grant everything to everybody"
  participant: "org.hyperledger.composer.system.Participant"
  operation: ALL
  resource: "org.hyperledger.composer.system.**"
  action: ALLOW
}
```

Everything permitted if this file is missing

Access Control Language

Rule controls permission to CRUD on Resource(s)

Simple Rule

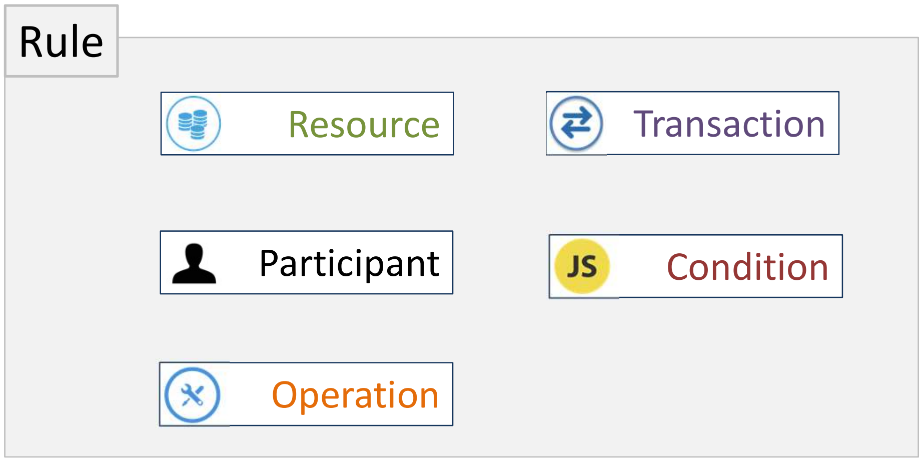
Controls access to namespace, asset or property of an asset by a participant type or participant instance

Conditional Rule

Boolean JavaScript expression evaluated at runtime to ALLOW or DENY access to the resource by the participant.

Access Control Language

Rule controls permission to CRUD on Resource(s)



Access Control Language

```
rule     Name-of-Rule {

         description:    " Provide description in quotes"

         resource:       "Fully qualified resource specification"

         participant:    "Fully qualified participant specification"

         operation:      ALL      *or*    Comma separated CRUD

         action:         ALLOW      *or*    DENY

}
```

Access Control Language

Resource

A Rule controls access to a Resource(s)

- Specific resource class

```
org.acme.airline.Aircraft
```

- Specific instance of class

```
org.acme.airline.Aircraft#CRAFT01
```

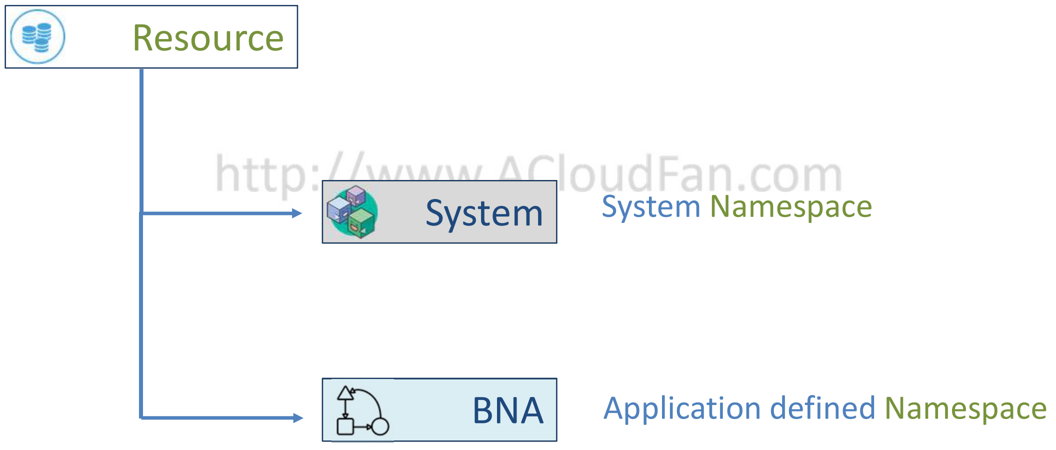
- All resource in Namespace

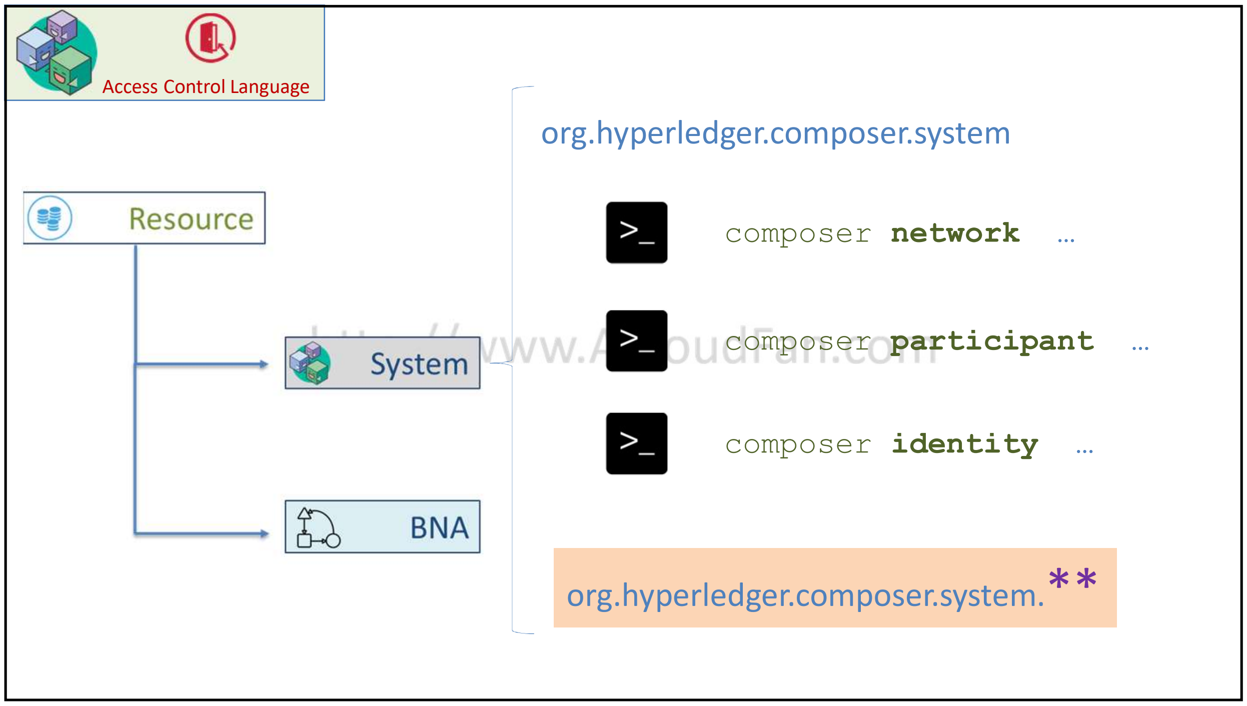
```
org.acme.airline.*
```

```
org.acme.airline.**
```

Recursive

Access Control Language

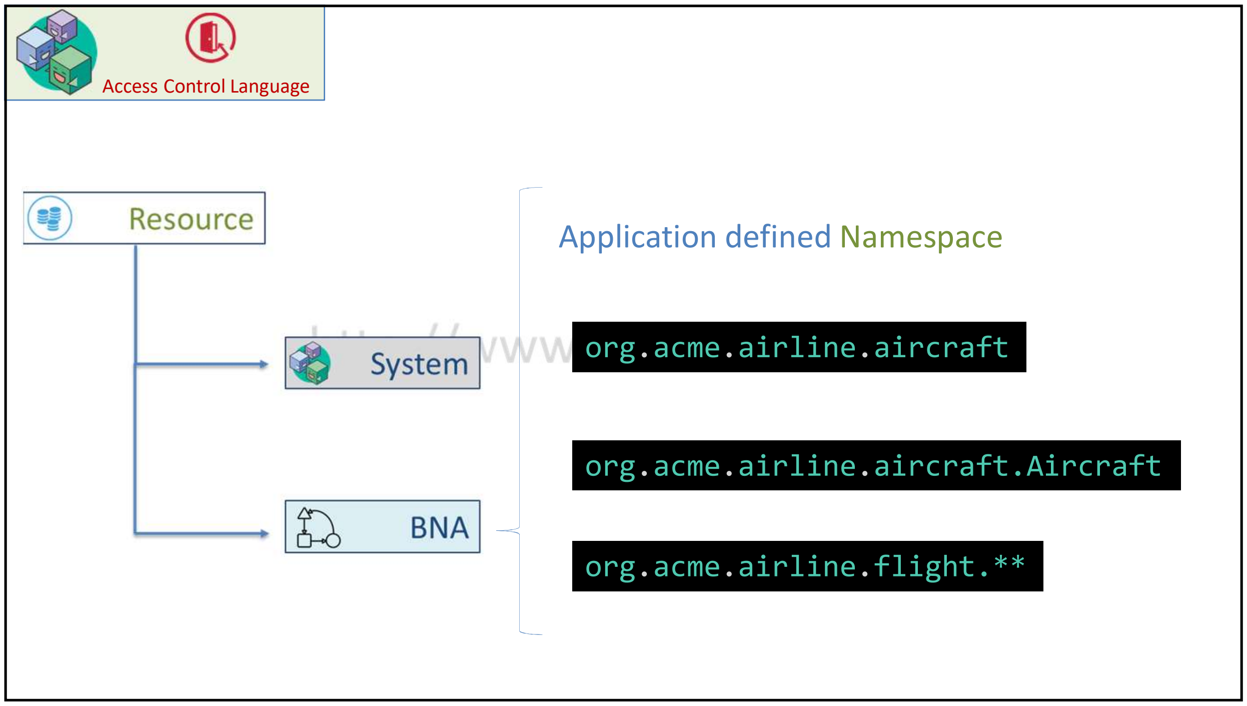




org.hyperledger.composer.system

```
>_ composer network ...
>_ composer participant ...
>_ composer identity ...
```

org.hyperledger.composer.system.**



Application defined Namespace

```
org.acme.airline.aircraft
org.acme.airline.aircraft.Aircraft
org.acme.airline.flight.**
```

Access Control Language

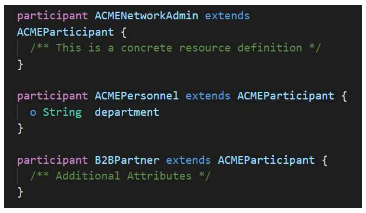
A Rule may be specific to a Participant type

- Keyword 'ANY' used if participant check is not required

Participant

- System org.hyperledger.composer.system.**NetworkAdmin**

- App Level



```
participant ACMENetworkAdmin extends
ACMEParticipant {
  /** This is a concrete resource definition */
}

participant ACMEPersonnel extends ACMEParticipant {
  o String  department
}

participant B2BPartner extends ACMEParticipant {
  /** Additional Attributes */
}
```

```
org.acme.airline.participant .ACMEPersonnel  #  johnd
```

Access Control Language

A Rule decides which Operation(s)

Operation

C R U D

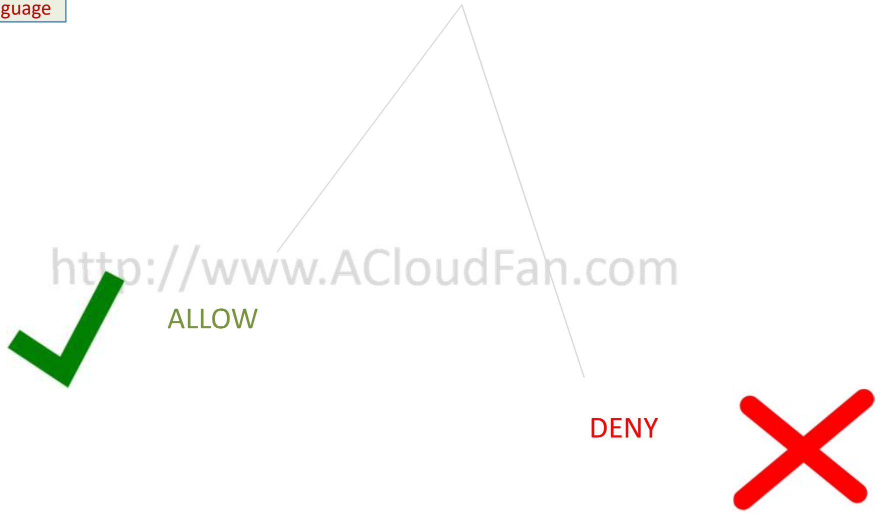
- Provide comma (,) separated list of ops

E.g., CREATE, READ, UPDATE

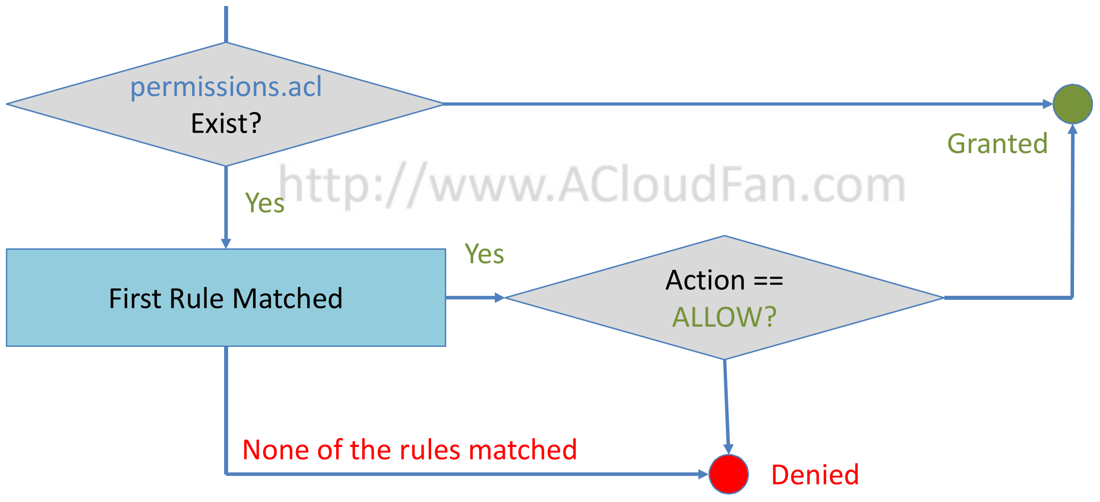
ALL

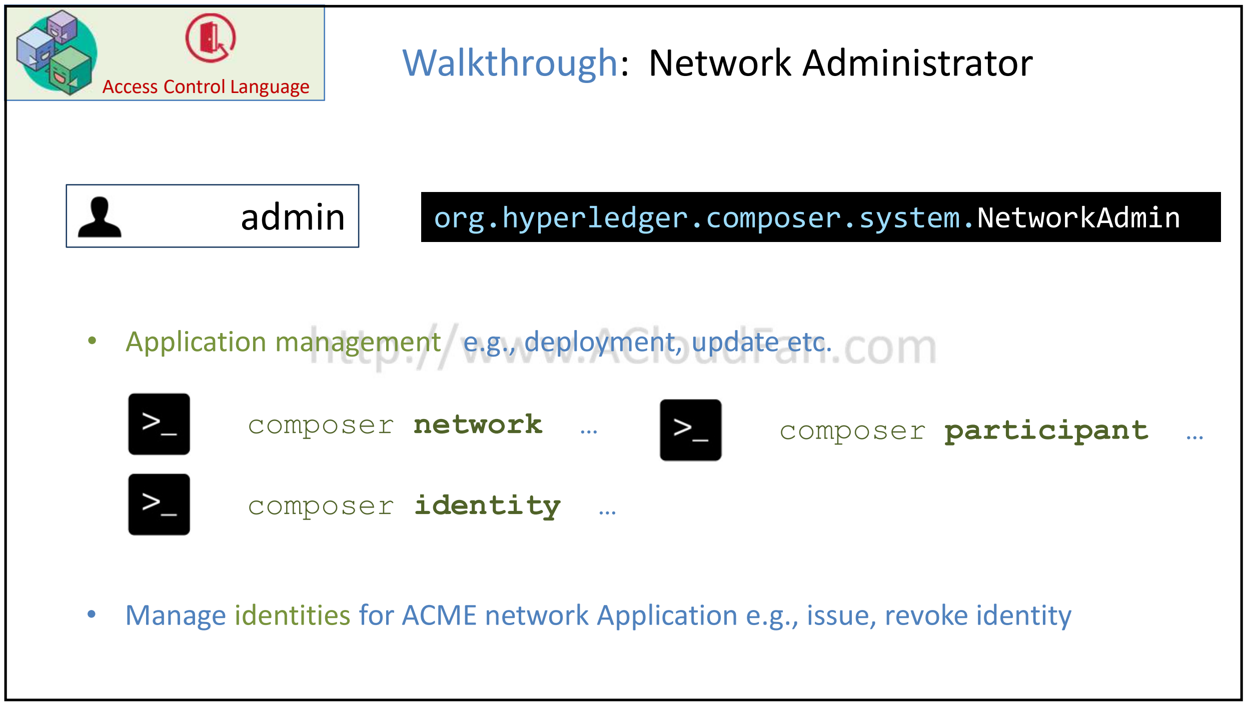
- Keyword for all operations

A Rule has an Action



Rules processing





Access Control Language

Walkthrough: Network Administrator

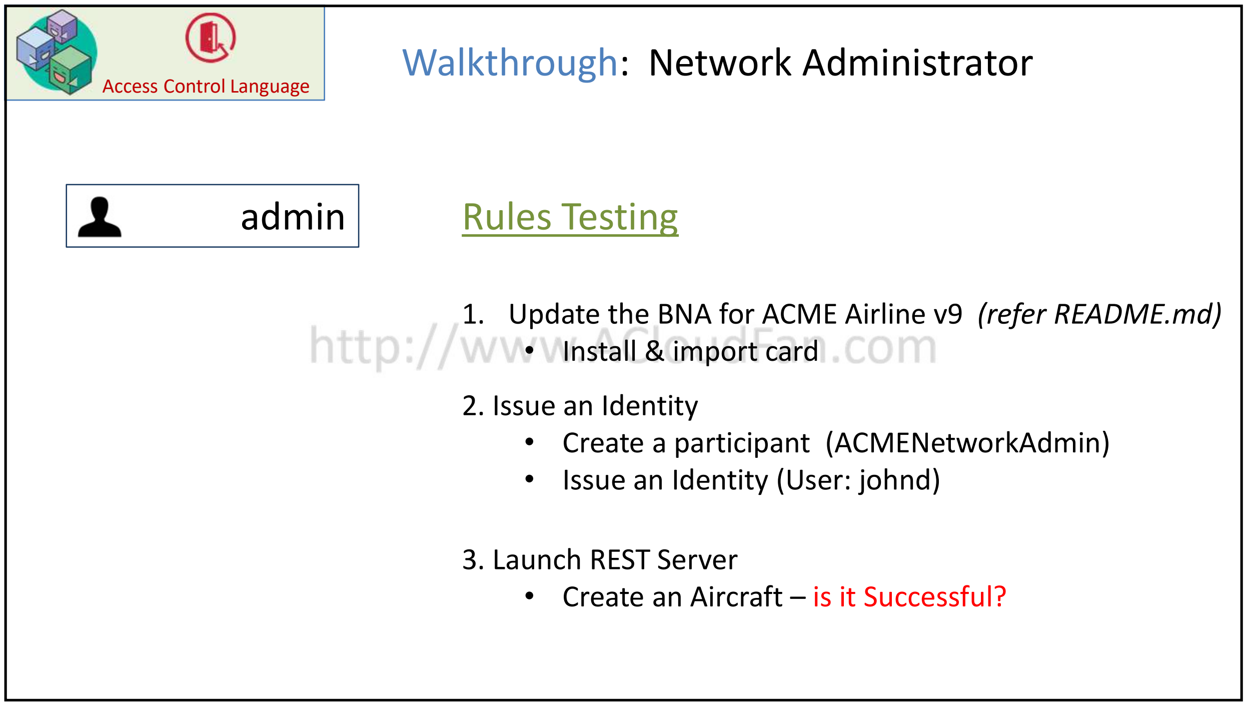
admin

```
org.hyperledger.composer.system.NetworkAdmin
```

- Application management e.g., deployment, update etc.

```
>_ composer network …
>_ composer participant …
>_ composer identity …
```

- Manage identities for ACME network Application e.g., issue, revoke identity



Access Control Language

Walkthrough: Network Administrator

admin

Rules Testing

1. Update the BNA for ACME Airline v9 *(refer README.md)*
   - Install & import card

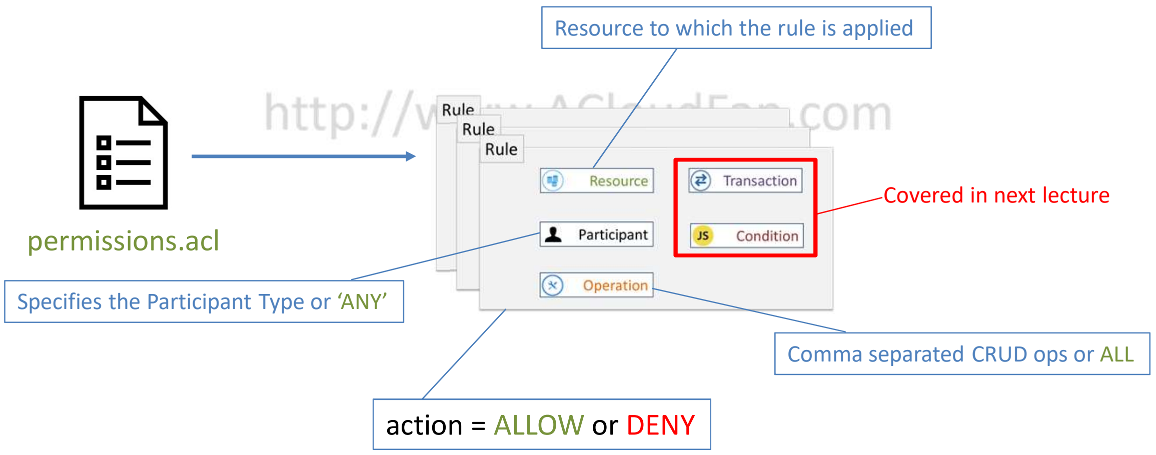
2. Issue an Identity
   - Create a participant (ACMENetworkAdmin)
   - Issue an Identity (User: johnd)

3. Launch REST Server
   - Create an Aircraft – is it Successful?

Summary

Access Control Language

- Declarative Access Control



Access Control & Permissions

Part 2 of 2 : Transaction access & Conditional Rule

raj@acloudfan.com

@acloudfan

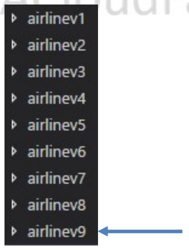
http://ACloudFan.com

Learning Objectives:

- Transaction & Condition element
- Walkthrough of a rule for Transaction

Incremental Creation of **ACME Air Domain Model**

https://github.com/acloudfan/**HLF-Course-Domain-Model**



```
airlinev1
airlinev2
airlinev3
airlinev4
airlinev5
airlinev6
airlinev7
airlinev8
airlinev9
```

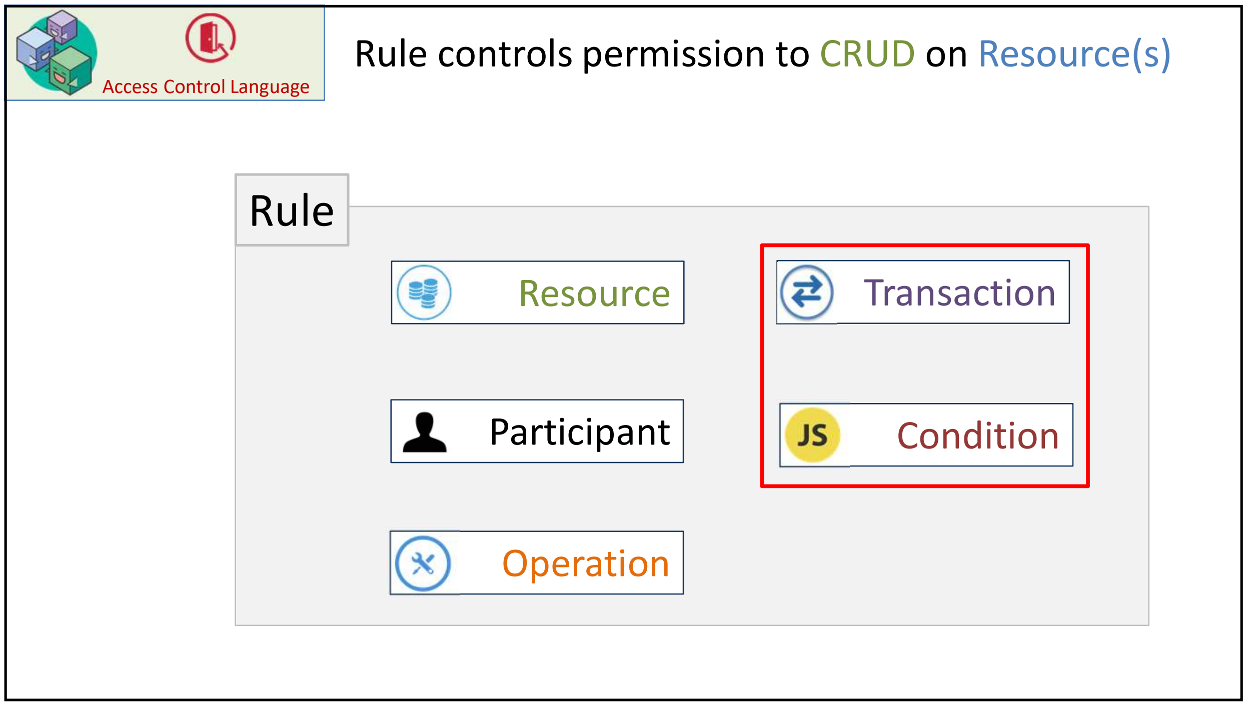
Code shown in video may change over time

Hyperledger Fabric API

https://github.com/acloudfan/**HLF-Fabric-API**

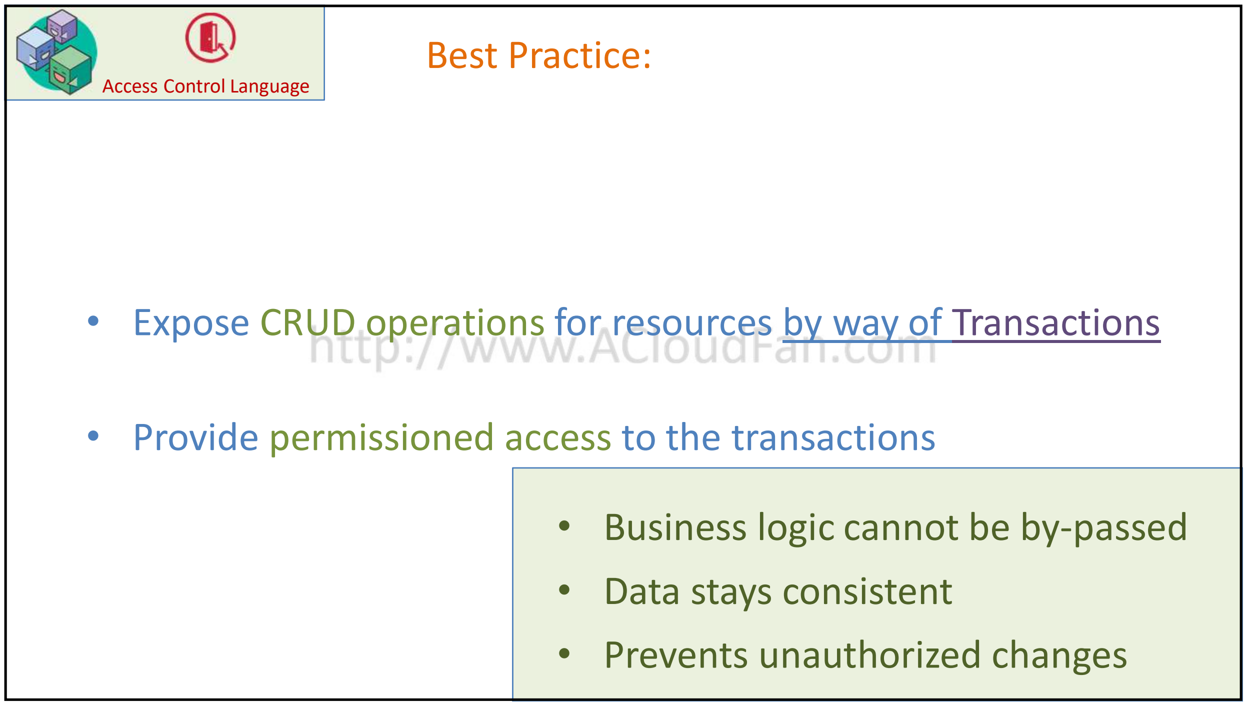
JS bn-test-permissions.js

Code shown in video may change over time



Access Control Language

Rule controls permission to CRUD on Resource(s)



Access Control Language

Best Practice:

- Expose CRUD operations for resources by way of Transactions
- Provide permissioned access to the transactions

- Business logic cannot be by-passed
- Data stays consistent
- Prevents unauthorized changes

Access Control Language

Transaction

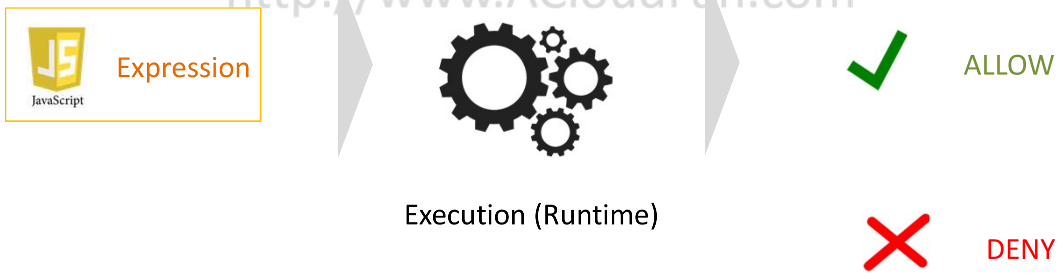
- CreateFlight Transaction Adds a new Flight instance to registry

Resource	`org.acme.airline.flight.Flight`
Operation	CREATE
Transaction	`org.acme.airline.flight.CreateFlight`

Access Control Language

Condition

- A valid Javascript conditional expression may be specified



Access Control Language

Condition

Symbolic access to the execution context

Resource	Operation	Participant	Transaction
(r)	(op)	(p)	(tx)

```
participant(p): "org.example.SampleParticipant"

resource(r): "org.example.SampleAsset"

condition: (r.owner.getIdentifier() == p.getIdentifier())
```

Access Control Language

Condition

- Complex conditions are supported
  - Utility function(s) in script file

```
participant(p): "org.example.SampleParticipant"

resource(r): "org.example.SampleAsset"

condition: (evaluateAccess(r, p))
```

Access Control Language

Walkthrough: ACMEPersonnel 'CreateFlight' Access

`org.acme.airline.participant.` ACMEPersonnel

William Smith (wills)

1. Create Access to HistorianRecord is needed for Transactions

2. Create Access needed for the CreateFlight transaction

Access Control Language

Walkthrough: ACMEPersonnel 'CreateFlight' Access

Rules Testing

1. Deploy/Update the BNA for ACME Airline v9 *(refer README.md)*

2. Issue an Identity
   - Create a participant (ACMEPersonnel)
   - Issue an Identity (User: wills)
   - Import the card for wills@airlinev9

wills

3. Use the utility to Test the setup

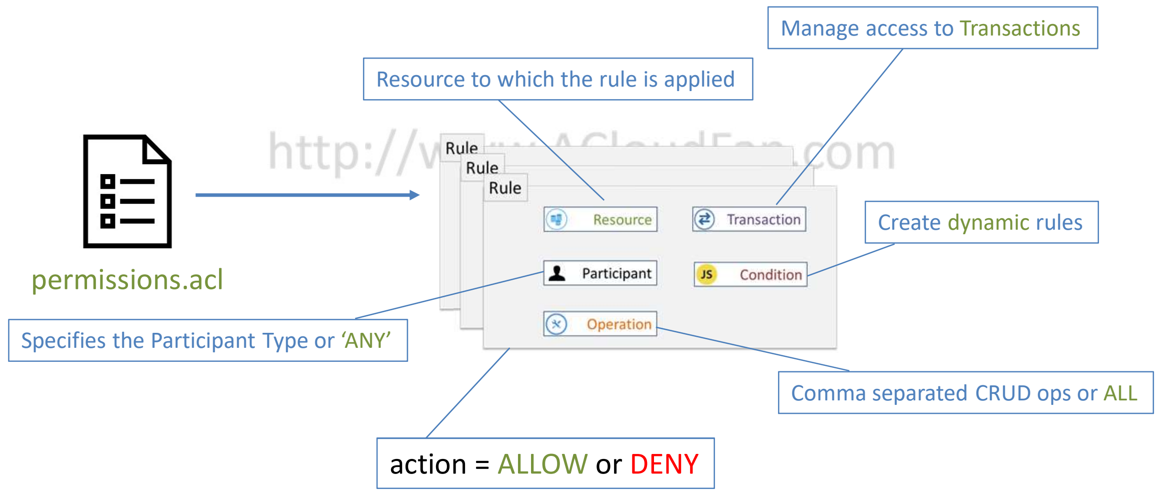
```
> node bn-test-permissions wills@airlinev9  <date in yyyy-mm-dd format>
```

Utility (a) Invokes **CREATE** on Flight Registry (b) Invokes the **CreateFlight** Transaction

Access Control Language

Summary

- Declarative Access Control


Manage access to Transactions
Resource to which the rule is applied
Rule
Rule
Rule
Resource
Transaction
Participant
Condition
Operation
Create dynamic rules
permissions.acl
Specifies the Participant Type or 'ANY'
Comma separated CRUD ops or ALL
action = ALLOW or DENY