CISCO

You make **possible**

How to build network stateful applications with Cisco ACI App Center

Nicolas Vermandé
Technical Marketing Engineer, IBNG

@nvermande

DEVNET-3576

Barcelona | January 27-31, 2020

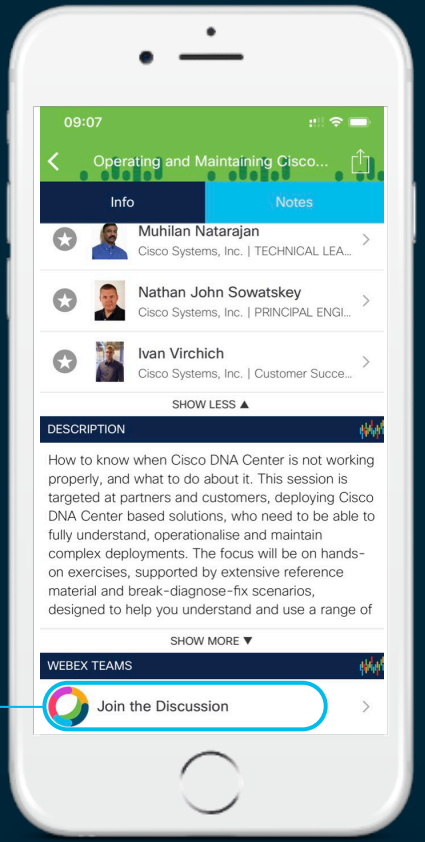
Cisco Webex Teams

Questions?

Use Cisco Webex Teams to chat with the speaker after the session

How

1. Find this session in the Cisco Events Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space



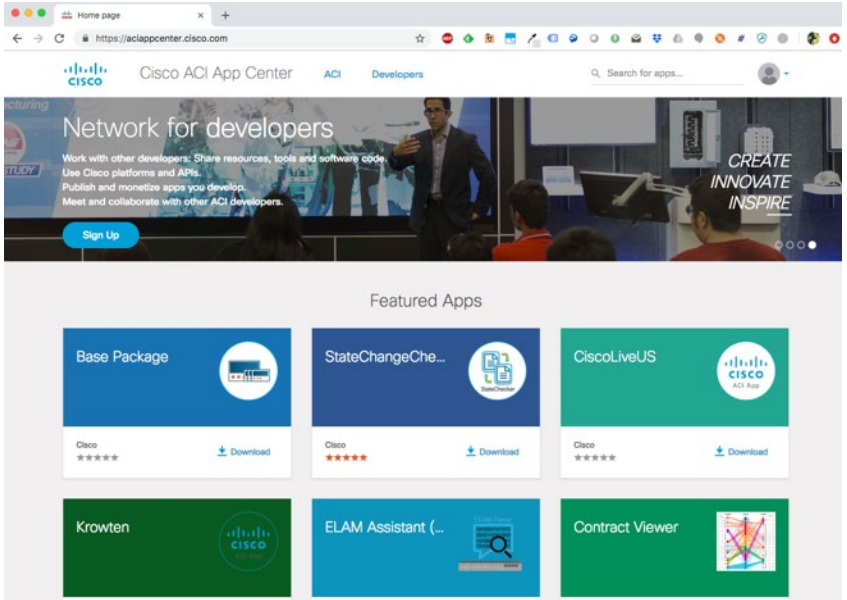
Objectives of this Session

- Understand the workflow to build a Stateful app
- Understand how to explore ACI object model
- How to build a simple HTTP server to handle routes and REST requests

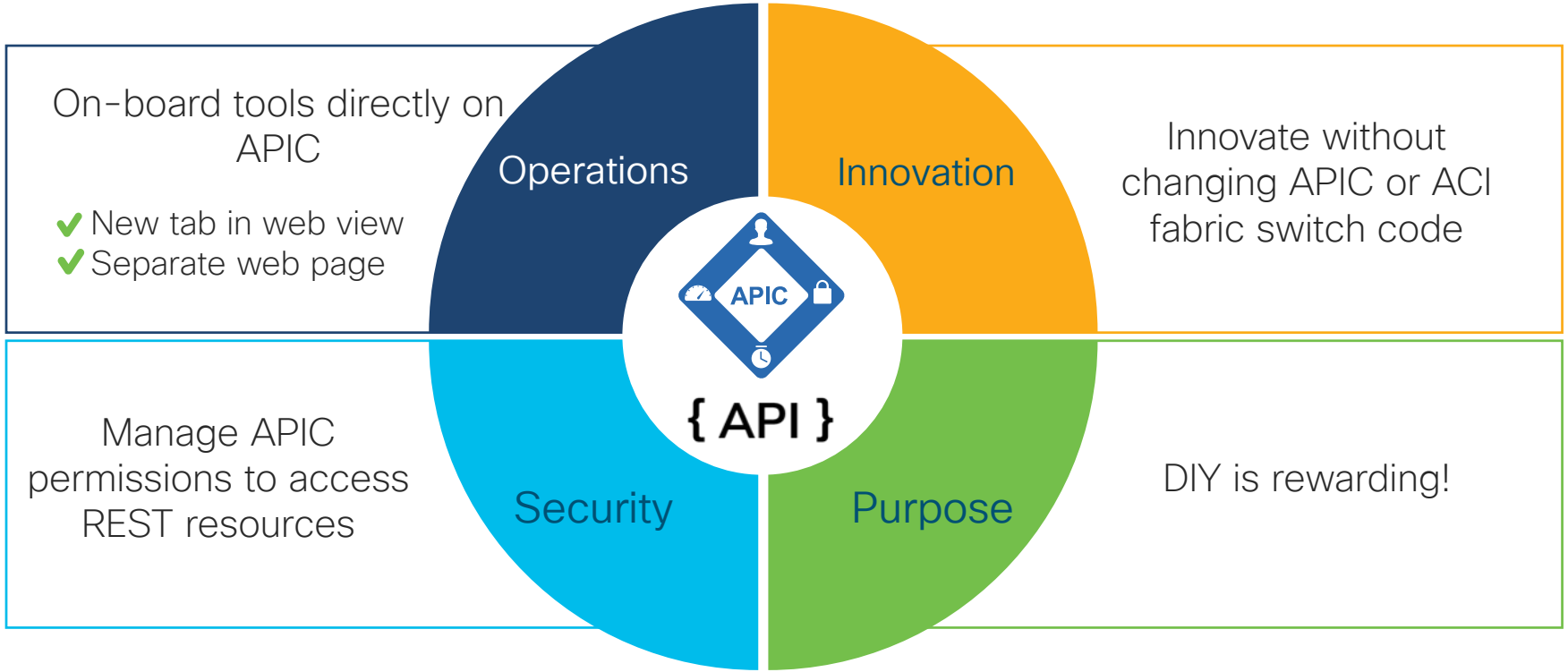
What is ACI App Center?

- Repository for user applications on APIC
- Cisco, ecosystem partner or customer driven
- A way to improve or customize ACI capabilities to your needs
- Based on any framework that can render HTML
- Use containers for stateful apps (backend)
- Browse for applications on ACI App Center website

https://aciappcenter.cisco.com



Why App Center

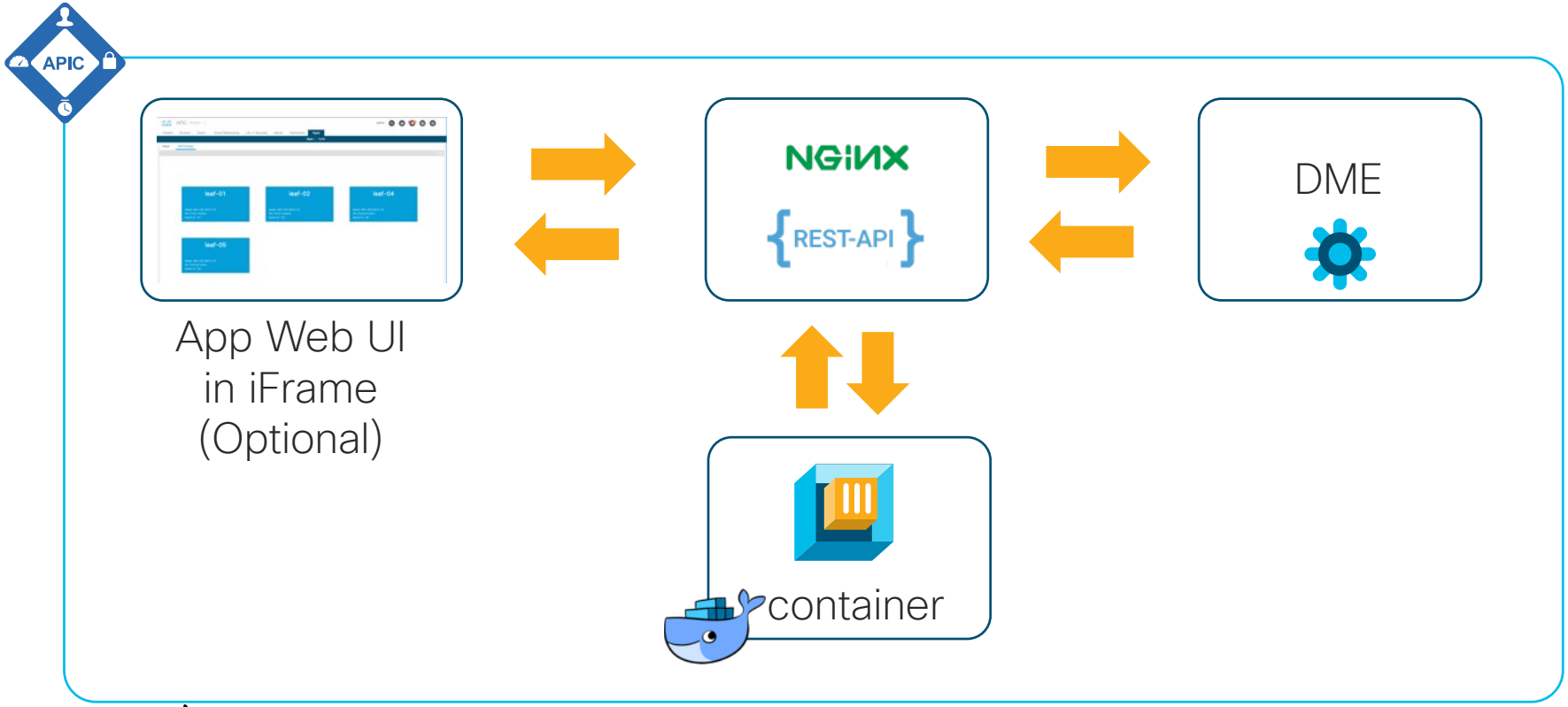


Support

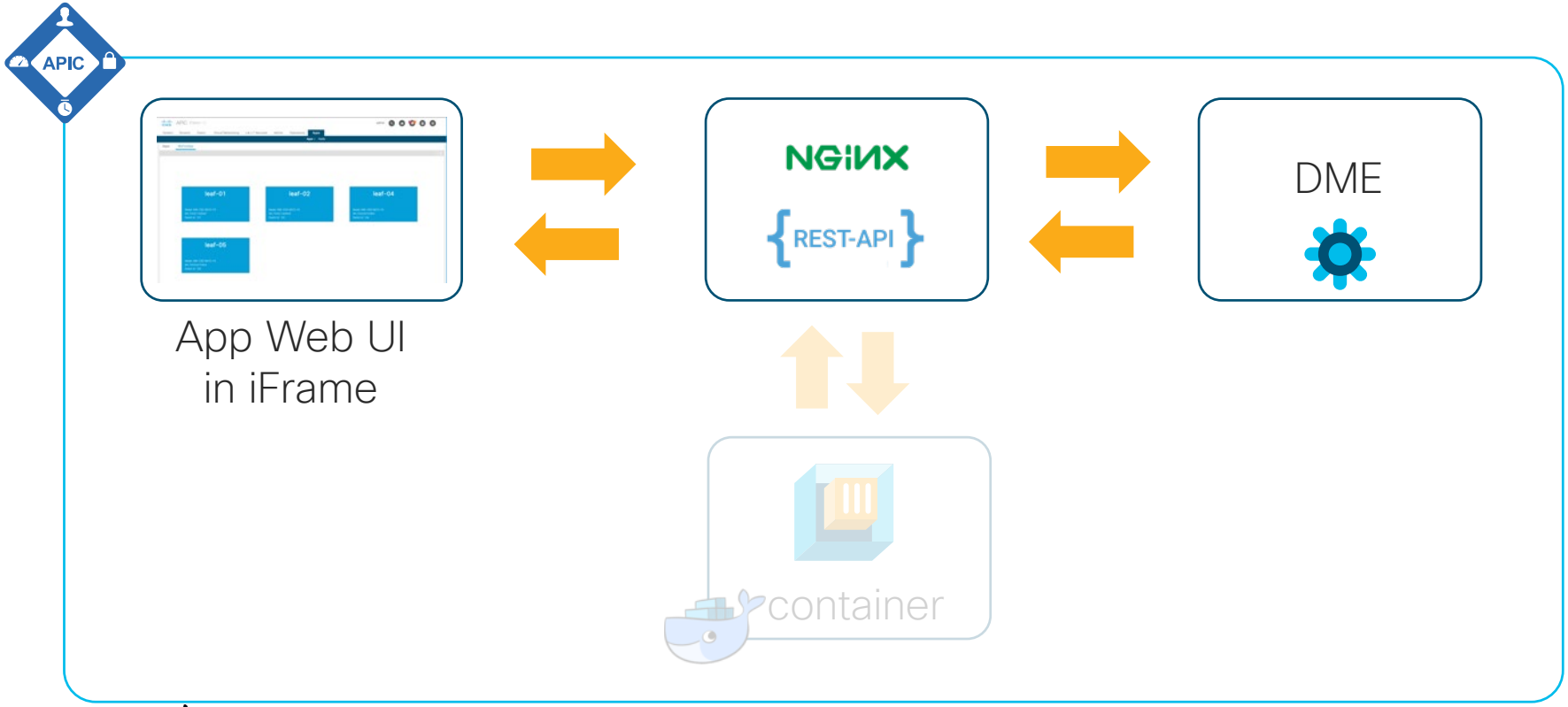
App developed by

Customer	Support provided by the community
Cisco	Support provided by TAC
Partner	Support provided by the partner

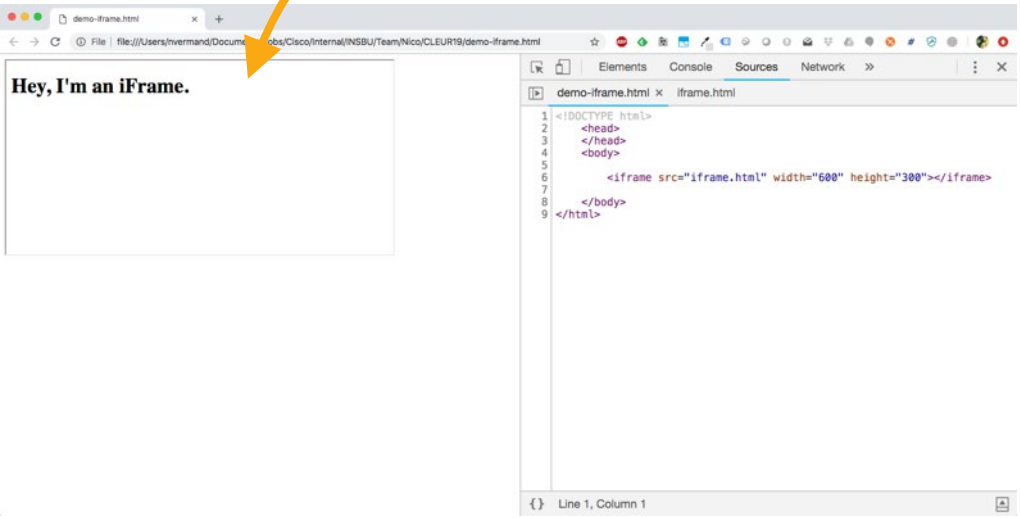
Stateful Architecture

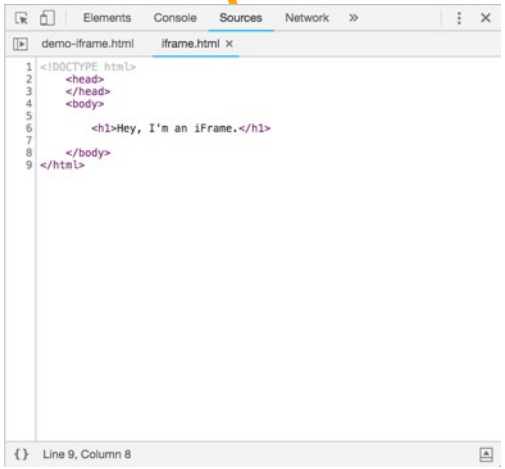


Stateless Architecture



iFrame





2 kinds of Applications:

- Stateless
- Stateful

This Session

Stateless App

- Active only when browsing App
- No data persists in the App
- Information is lost when browsing away from App tab or closing App
  - May need to be recomputed next time it is accessed
- Composed of **HTML**, **CSS** and **Javascript** files (and libraries)
- Interact with user through Web UI frontend
- When App is launched, SSO token is requested for current user and passed to the App

Stateful App

- Backend service running in docker container
- Runs on APIC shard leader for application Dn
- Filesystem is distributed (glusterFS) for persistent storage
- May have a stateless component for frontend UI
- Embedded HA with APIC cluster
- User API requests to backend are proxied through APIC at `/appcenter/<vendor>/<app_id>/<api_URI>`
- Certificates are used for authentication and special aaaUser is created with permissions defined in APP configuration file

App Deployment and Distribution

- App is packaged with provided tool
- 2 Distribution approaches
  - Upload App to APIC directly (`.aci` packaged file)
  - Publish to Cisco App Center website
    - Needs Developer private key to sign app
    - Upload on website
- Admin can enable signature validation on APIC
  - Only Cisco signed app will be allowed to install

Stateful App 101

How to build a Stateful App

https://developer.cisco.com/docs/aci/#!app-center-resources/getting-started-with-aci-app-center

Create App environment

Download `cisco_aci_app_tools-1.1.tar.gz` from DevNet. It contains all tools to bootstrap your application

```
$ pip install cisco_aci_app_tools-1.1.tar.gz
```

Use `aci_app_creator.py` to initialize your App directory structure and populate required metadata

Build Container Image

The `Image` folder contains the the .tgz image for the application. Images cannot be shared across multiple apps.

The `Service` folder contains the `start.sh` **script that is executed when the container starts and initialize your application**

How to build a Stateful App

Develop your App

Use your favorite language to program the app backend. A plethora of frameworks are available, such as

- Django, Flask (python)
- Ruby on Rails
- Martini, Gin Gonic (!!!!), Gorilla (Golang)
- node.js, etc.
- Add frontend if desired

Package your App

Update your App to use provided token for user authentication

Package your App with `aci_app_packager.py`. You can also validate your App beforehand with `aci_app_validator.py`

Optionally, publish your app on App Center website (App Store)

How to build a Stateless App

Install your App

Upload you packaged App to APIC from your development environment or from the App Center website

Enable the Application

Initializing the environment

- `cisco_aci_app_tools-1.1.tar.gz` provides the following scripts (under the `tools` folder):
  - `aci_app_creator.py`
  - `aci_app_packager.py`
  - `aci_app_validator.py`
- It also contains stateful and stateless app Templates that are used to provision folder structure and populate metadata

```
$ python aci_app_creator.py
*******************************************
*              ACI App Creator            *
*******************************************
Welcome! This tool will guide you through the creation of a fully
functional ACI App Center application.
The information that you will provide can be changed later on, please
read the ACI App Center Developer Guide to learn more about it.
```

Initializing the environment

```
myApp
    ├── App.json
    ├── Legal
    │   ├── Cisco_..._Agreement.docx
    │   └── Cisco_...Questionnaire.docx
    ├── Media
    │   ├── IntroVideo
    │   │   └── AppCenter Video Placeholder.mp4
    │   ├── Readme
    │   │   └── Cisco_App_Center_License.txt
    │   ├── License
    │   │   └── readme.txt
    │   └── Snapshots
    │       └── snapshot.png
    ├── UIAssets
    │   ├── app.html
    │   ├── app-start.html
    │   └── icon.png
    ├── Image
    │   aci_appcenter_docker_image.tgz
    └── Service
        server.py
        start.sh
```

(Optional) List of videos introducing the application

(Optional) Application preview snapshots

Application thumbnail

Initializing the environment

```
myApp
    ├── App.json
    ├── Legal
    │   ├── Cisco_..._Agreement.docx
    │   └── Cisco_...Questionnaire.docx
    ├── Media
    │   ├── IntroVideo
    │   │   └── AppCenter Video Placeholder.mp4
    │   ├── Readme
    │   │   └── Cisco_App_Center_License.txt
    │   ├── License
    │   │   └── readme.txt
    │   └── Snapshots
    │       └── snapshot.png
    ├── UIAssets
    │   ├── app.html
    │   ├── app-start.html
    │   └── icon.png
    ├── Image
    │   aci_appcenter_do
    ├── Service
    │   server.py
    │   start.sh
```

Sample Flask application to run the webserver that will serve the REST API requests sent to the container and proxied by APIC. It is typically started by the `start.sh` file.

Initializing the environment

```
myApp
    ├── App.json
    ├── Legal
    │   ├── Cisco_..._Agreement.docx
    │   └── Cisco_...Questionnaire.docx
    ├── Media
    │   ├── IntroVideo
    │   │   └── AppCenter Video Placeholder.mp4
    │   ├── Readme
    │   │   └── Cisco_App_Center_License.txt
    │   ├── License
    │   │   └── readme.txt
    │   └── Snapshots
    │       └── snapshot.pn
    ├── UIAssets
    │   ├── app.html
    │   ├── app-start.html
    │   └── icon.png
    ├── Image
    │   aci_appcenter_d
    └── Service
        server.py
        start.sh
```

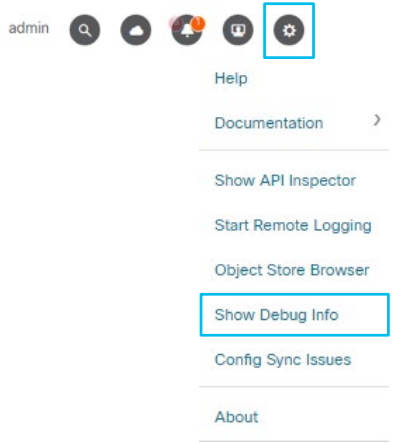
Script that builds the logic to start all required services that compose your application.
It must contain the line that start the webserver and optionally start other components that are co-located under the Service folder. This includes database, extra configuration, initialization steps and tests, etc.

Initializing the environment

```
myApp
    App.json
    Legal
        Cisco_..._Agreement.docx
        Cisco_...Questionnaire.docx
    Media
        IntroVideo
            AppCenter Video Placeholder.mp4
        Readme
            Cisco_App_Center_License.txt
        License
            readme.txt
        Snapshots
            snapshot.png
    UIAssets
        app.html
        app-start.html
    css
        myApp.css
    Fonts
    scripts
        myApp.js
        jquery-3.1.1.min.js
```

```
{
    "apicversion":"2.2(1n)",
    "appid":"MyFirstApp",
    "author":"nvermand",
    "category":[
        "Tools and Utilities"
    ],
    "contact":{
        "contact-email":"nvermand@cisco.com",
        "contact-phone":"123-4567890",
        "contact-url":"http://www.cisco.com/go/aci"
    },
    "iconfile":"icon.png",
    "insertionURL":"fv:infoTenant:center",
    "metaversion":"1.1",
    "name":"MyFirstApp",
    "permissions":[
        "admin"
    ],
    "permissionslevel":"read",
    "shortdescr":"This is my first App",
    "vendor":"Cisco",
    "vendordomain":"Cisco",
    "version":"1.0"
}
```

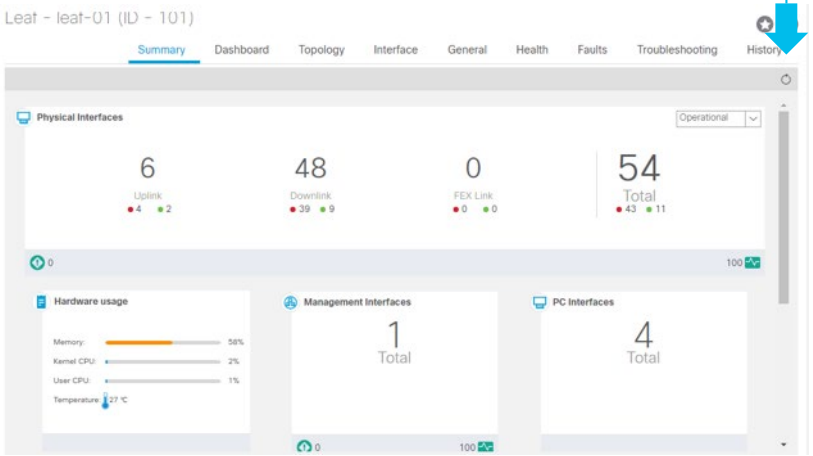
Initializing the environment



Enable Debug Info to show layout information

Current Screen: insieme.stromboli.layout.Tab [eqpt:infoLeaf:center:a00] | Current Mo: insieme.stromboli.model.def.fabricNode [topology/pod-1/node-101]

Initializing the environment



Current Screen: insieme.stromboli.layout.Tab [eqpt:infoLeaf:center:a00] | Current Mo: insieme.stromboli.model.def.fabricNode [topology/pod-1/node-101]

```
insertionURL: "eqpt:infoLeaf:center"
```

Initializing the environment

```
myApp
    App.json
    Legal
        Cisco_..._Agreement.docx
        Cisco_...Questionnaire.docx
    Media
        IntroVideo
            AppCenter Video Placeholder.mp4
        Readme
            Cisco_App_Center_License.txt
        License
            readme.txt
        Snapshots
            snapshot.png
    UIAssets
        app.html
        app-start.html
    css
        myApp.css
    Fonts
    scripts
        myApp.js
        jquery-3.1.1.min.js
```

"app_Cisco_MyFirstApp_token"

```
<script type="text/javascript">
function onBodyLoad() {
    var arr, url, newUrl = "app.html", myMask = new Ext.LoadMask(Ext.getBody(),
{msg:"Please wait..."});
/...
var tokenObj = Ext.decode(e.data, true);
if (tokenObj) {
    Ext.util.Cookies.set('app_' + tokenObj.appId + '_token', tokenObj.token);
    Ext.util.Cookies.set('app_' + tokenObj.appId + '_urlToken', tokenObj.urlToken);
/...
```

Initializing the environment

```
myApp
    App.json
    Legal
        Cisco_..._Agreement.docx
        Cisco_...Questionnaire.docx
    Media
        IntroVideo
            AppCenter Video Placeholder.mp4
        Readme
            Cisco_App_Center_License.txt
        License
            readme.txt
        Snapshots
            snapshot.png
    UIAssets
        app.html
        app-start.html
    css
        myApp.css
    Fonts
    scripts
        myApp.js
        jquery-3.1.1.min.js
```

app.html is called from app-start.html

```
<script type="text/javascript">
function onBodyLoad() {
    var arr, url, newUrl = "app.html", myMask = new Ext.LoadMask(Ext.getBody(),
{msg:"Please wait..."});
/...
var tokenObj = Ext.decode(e.data, true);
if (tokenObj) {
    Ext.util.Cookies.set('app_' + tokenObj.appId + '_token', tokenObj.token);
    Ext.util.Cookies.set('app_' + tokenObj.appId + '_urlToken', tokenObj.urlToken);
/...
```

Initializing the environment

```
myApp
    ├── App.json
    ├── Legal
    │   ├── Cisco_..._Agreement.docx
    │   └── Cisco_...Questionnaire.docx
    ├── Media
    │   ├── IntroVideo
    │   │   └── AppCenter Video Placeholder.mp4
    │   ├── Readme
    │   │   └── Cisco_App_Center_License.txt
    │   ├── License
    │   │   └── readme.txt
    │   └── Snapshots
    │       └── snapshot.png
    ├── UIAssets
    │   ├── app.html
    │   └── app-start.html
    ├── css
    │   └── myApp.css
    ├── Fonts
    └── scripts
        ├── myApp.js
        └── jquery-3.1.1.min.js
```

Token must be retrieved every time APIC refreshes it and sends it to the application.

```
<script type="text/javascript">
/...
    window.addEventListener('message', function (e) {
        if (e.source === window.parent) {
    var tokenObj = Ext.decode(e.data, true);
        if (tokenObj) {
            Ext.util.Cookies.set("app_Cisco_MyFirstApp_token", tokenObj.token);
            Ext.util.Cookies.set("app_Cisco_MyFirstApp_urlToken", tokenObj.urlToken);
        }
/...
```

Either by calling the function that updates the cookie

Initializing the environment

```
myApp
    App.json
    Legal
        Cisco_..._Agreement.docx
        Cisco_...Questionnaire.docx
    Media
        IntroVideo
            AppCenter Video Placeholder.mp4
        Readme
            Cisco_App_Center_License.txt
        License
            readme.txt
        Snapshots
            snapshot.png
    UIAssets
        app.html
        app-start.html
    css
        myApp.css
    Fonts
    scripts
        myApp.js
        jquery-3.1.1.min.js
```

Token must be retrieved every time APIC refreshes it and sends it to the application.

```
<script type="text/javascript">
    window.APIC_DEV_COOKIE = Ext.util.Cookies.get("app_Cisco_MyFirstApp_token");
    window.APIC_URL_TOKEN =  Ext.util.Cookies.get("app_Cisco_MyFirstApp_urlToken");
/...
    window.addEventListener('message', function (e) {
        if (e.source === window.parent) {
            var tokenObj = Ext.decode(e.data, true);
            if (tokenObj) {
                window.APIC_DEV_COOKIE = tokenObj.token;
                window.APIC_URL_TOKEN = tokenObj.urlToken;
            }
/...
```

Or using a global variable that can be reused later

Initializing the environment

```
myApp
    ├── App.json
    ├── Legal
    │   ├── Cisco_..._Agreement.docx
    │   └── Cisco_...Questionnaire.docx
    ├── Media
    │   ├── IntroVideo
    │   │   └── AppCenter Video Placeholder.mp4
    │   ├── Readme
    │   │   └── Cisco_App_Center_License.txt
    │   ├── License
    │   │   └── readme.txt
    │   └── Snapshots
    │       └── snapshot.png
    ├── UIAssets
    │   ├── app.html
    │   └── app-start.html
    ├── css
    │   └── myApp.css
    ├── Fonts
    └── scripts
        ├── myApp.js
        └── jquery-3.1.1.min.js
```

This is your App Javascript code

Initializing the environment

```
myApp
    App.json
    Legal
        Cisco_..._Agreement.docx
        Cisco_...Questionnaire.docx
    Media
        IntroVideo
            AppCenter Video Placeholder.mp4
        Readme
            Cisco_App_Center_License.txt
        License
            readme.txt
        Snapshots
            snapshot.png
    UIAssets
        app.html
        app-start.html
    css
        myApp.css
    Fonts
    scripts
        myApp.js
        jquery-3.1.1.min.js
```

Add you CSS files

Add your libraries locally

TIP

Minify your libraries to reduce their size

APIC and Backend Service

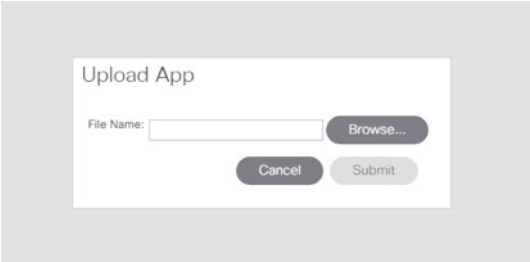
- Application User RBAC is enforced in NGINX
- At installation time, public and private keys are generated and associated with the App user
- App data is clustered across all APICs in the cluster
- When a node goes down, a new leader is elected to start the application
- All code under Service folder is mounted to /home/app/src inside the container.

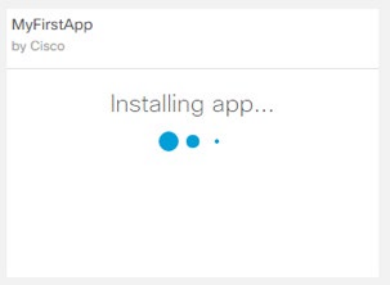
Package and Install your App

- Option 1: Direct upload to APIC

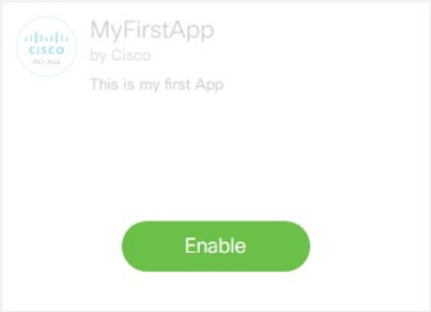
```
$python aci_app_packager.py -f ~/sources/Cisco_MyFirstApp/
```

Upload `.aci` file to APIC in App Center





Enable App



Package and Install your App

- Option 2: Let's publish our App to the App Store

Create a developer account

Generate developer private key

Package App signed with private key

```
$python aci_app_packager.py -f ~/sources/Cisco_MyFirstApp/ -p /home/nvermand/aci_app.pem
```

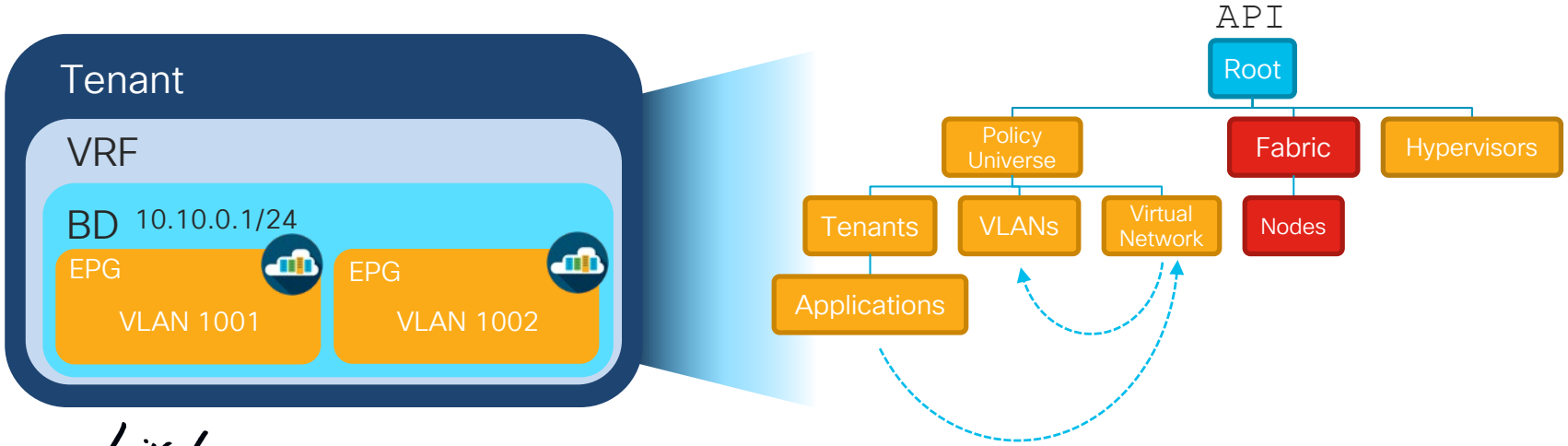
Upload App to App Store for approval

Upload and enable App on APIC

Working with the ACI Object Model

ACI Object Model

- ACI has a modelled representation of everything APIC knows
- ACI object model is a distributed MIT (Management Information Tree) structure, fully accessible through REST API
- Every node is a managed object (MO) with class, attributes and a distinguished name (Dn)



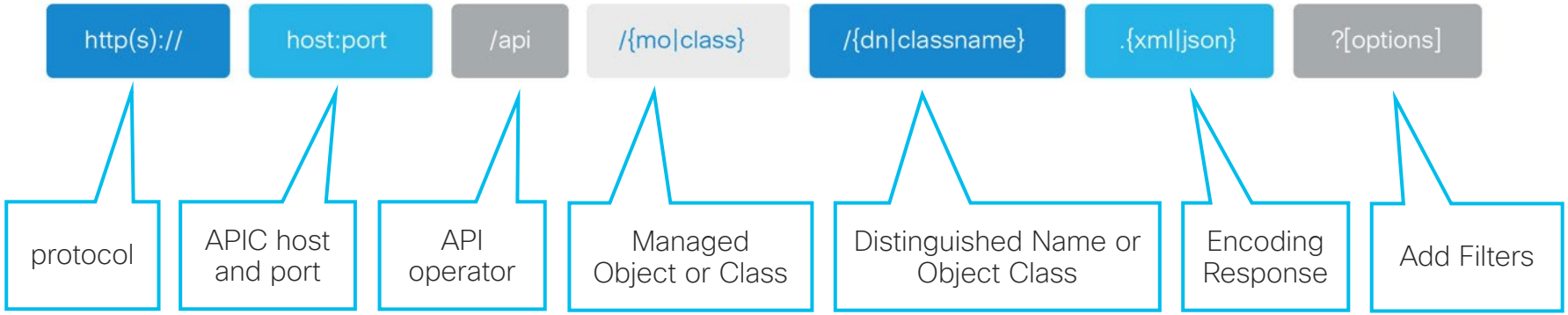
REST API principles with APIC

HTTP Verb	Action	Behavior
GET	Read	Nullipotent
POST	Create/Update	Idempotent
DELETE	Delete	Idempotent

- Extra status attribute used in ACI
  - “Created”
  - “Deleted”
- POST request to existing resource with status: “created” will trigger HTTP error 400 (Bad Request)
- POST request to existing resource with status “deleted” will delete the resource

- Payloads can be either XML or JSON
  - Specified by the file extension in URI
  - Content-Type and Accept header is ignored
- Within object payload, attributes entries must be defined before children entries
- Attributes can’t be omitted
  - Use “`attributes”: {}` when needed

How to build the URI?



Working with Filters

Filter Type	Syntax	Description
query-target	{ self | children | subtree }	Scope of the query
target-subtree-class	class name (e.g. fvTenant)	Respond only elements including the specified class
query-target-filter	filter expression (e.g. eq(fv.Tenant.name,”myTenant”)	Respond only elements matching conditions
rsp-subtree	{ no | children | full }	Specifies child object level included in the response
rsp-subtree-class	class name	Respond only specified class
rsp-subtree-filter	filter expressions	Respond only classes matching conditions
rsp-subree-include	{ faults | health | stats | ...}	Request additional objects
order-by	classname.propery | { asc | desc }	Sort the response based on the property values

Example

Find EPGs in ANP ”app-1” for all Tenants that have this ANP defined

```
https://<APIC>/api/class/fvTenant.json
```

This is a class query

Example

Find EPGs in ANP ”app-1” for all Tenants that have this ANP defined

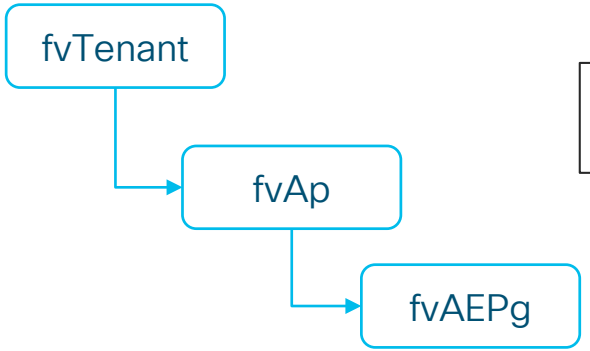
```
https://<APIC>/api/class/fvTenant.json
```

Start with Tenant as the top object

Example

Find EPGs in ANP ”app-1” for all Tenants that have this ANP defined

```
https://<APIC>/api/class/fvTenant.json
```

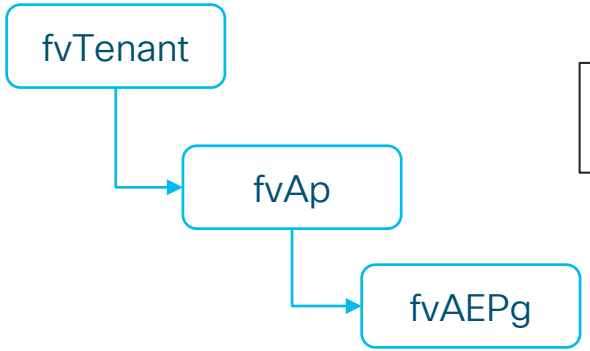


fvAp is a child of fvTenant, so we must extend the scope of the query

Example

Find EPGs in ANP ”app-1” for all Tenants that have this ANP defined

```
https://<APIC>/api/class/fvTenant.json?query-target=children
```

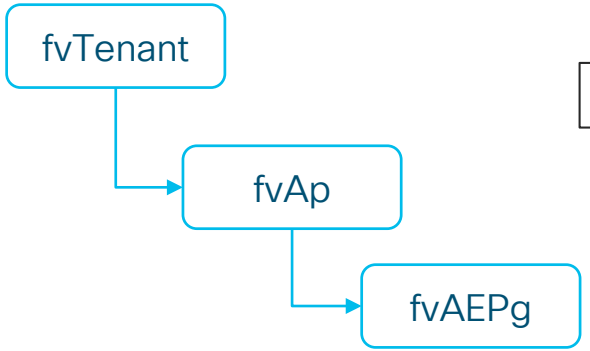


fvAp is a child of fvTenant, so we must extend the scope of the query

Example

Find EPGs in ANP ”app-1” for all Tenants that have this ANP defined

```
https://<APIC>/api/class/fvTenant.json?query-target=children
                                &query-target-filter=eq(fvAp.name,”app-1”)
```

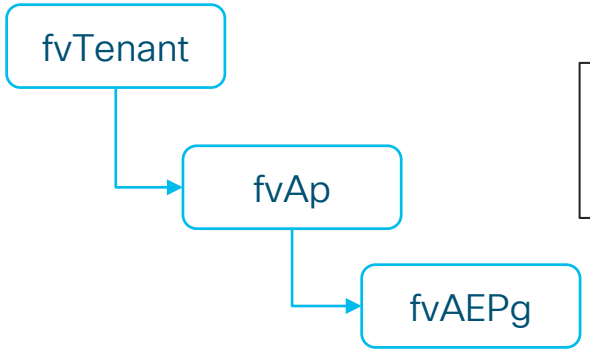


We must filter on fvAp name

Example

Find EPGs in ANP ”app-1” for all Tenants that have this ANP defined

```
https://<APIC>/api/class/fvTenant.json?query-target=children
                        &query-target-filter=eq(fvAp.name,”app-1”)
                        &rsp-subtree=children
```



Response must include EPG information. fvAEPg is a child of fvAp, so response must include children information

What do I do with HTTP response payload?

- Once the query has been defined, HTTP response payload must be handled
- It will contain comprehensive information
- Using native Javascript, it is then very easy to:
  - Check HTTP response status
  - Parse the data we need
  - Create Javascript objects with this information

How to start exploring?

- in APIC, right-click object, save-as JSON file
- Use Visore: http://<APIC>/visore.html to browse the object model
- Use the API inspector in APIC
- Use browser developer tools console

Time to build the REST API for the backend!

Fancy doing this in Go?

What is Go?

- Programming language created at Google by Robert Griesemer, Rob Pike (Unix), and Ken Thompson (Unix, C).
- Statically compiled into machine code. Produced binary doesn't need any dynamic library or runtime environment
- Static Typing. Types are checked before execution time
- Concurrency (!=Parallelism). Pipelines that makes efficient use of CPUs and I/O
- Powerful library (import directly from github)
- Composition, not inheritance. About behavior rather than data.

HTTP handler

The famous Handler interface:

```
//Handler interface
type Handler interface {
  ServeHTTP(ResponseWriter, *Request)
}
```

Any type that satisfies the Handler interface is of type Handler

NET/HTTP package – http.Handle()

https://golang.org/pkg/net/http/

```go
package main

import (
    "log"
    "net/http"
)

type server struct{}

func (s *server) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    w.WriteHeader(http.StatusOK)
    w.Header().Set("Content-Type", "application/json")
    w.Write([]byte(`{"message": "hello world"}`))
}

func main() {
    s := &server{}
    http.Handle("/", s)
    log.Println("Listening...")
    log.Fatal(http.ListenAndServe(":8080", nil))
}
```

Function receiver Allow for doted notation `s.ServeHTTP()`

Second argument is a Handler

Wait...why do I need to create a struct if the only thing it has is a method???

NET/HTTP package – http.HandleFunc()

```
package main

import (
  "log"
  "net/http"
)

func home(w http.ResponseWriter, r
*http.Request) {
  w.WriteHeader(http.StatusOK)
    w.Header().Set("Content-Type",
"application/json")
    w.Write([]byte(`{"message": "hello
world"}`))
}

func main() {
  http.HandleFunc("/", home)
 log.Println("Listening...")
  log.Fatal(http.ListenAndServe(":8080", nil))
}
```

Not type, so no receiver

Second argument is a function

Add some more HTTP verbs

```
func home(w http.ResponseWriter, r *http.Request) {
  w.Header().Set("Content-Type", "application/json")
  switch r.Method {
  case "GET":
    w.WriteHeader(http.StatusOK)
    w.Write([]byte(`{"message": "GET was used"}`))
  case "POST":
    w.WriteHeader(http.StatusCreated)
    w.Write([]byte(`{"message": "POST was used"}`))
  case "PUT":
    w.WriteHeader(http.StatusAccepted)
    w.Write([]byte(`{"message": "PUT was used}`))
  case "DELETE":
    w.WriteHeader(http.StatusOK)
    w.Write([]byte(`{"message": "DELETE was used"}`))
  default:
    w.WriteHeader(http.StatusNotFound)
    w.Write([]byte(`{"message": "NOT FOUND"}`))
  }
}
```

Test the code

```
$ curl -X POST localhost:8080
{"message": "POST was used"}

$ curl -X DELETE localhost:8080
{"message": "DELETE was used"}

$ curl -X PUT localhost:8080
{"message": "PUT was used}
```

Create your Router (API not Network!)

aka http.ServeMux https://golang.org/pkg/net/http/#ServeMux

- Return the current time when serving API Endpoint `/time`
- Redirect to developer.cisco.com when serving API Endpoint /info

```go
package main

import (
  "log"
  "net/http"
  "time"
)

type timeHandler struct {
  format string
}

func (th *timeHandler) ServeHTTP(w http.ResponseWriter, r *http.Request) {
  tn := time.Now().Format(th.format)
  w.Write([]byte("The time is: " + tn))
}
```

Implements `ServeHTTP()`, therefore `timeHandler` is of type Handler

```go
func main() {
  mux := http.NewServeMux()

  rh := http.RedirectHandler("http://developer.cisco.com", 307)
  mux.Handle("/info", rh)

  th := &timeHandler{format: time.RFC1123}
  mux.Handle("/time", th)

  log.Println("Listening...")
  http.ListenAndServe(":8080", mux)
}
```

Test the code

```
$ curl -L localhost:8080/info | head -n 5
 <!DOCTYPE html>
 <html lang="en">
 <head>
     <meta charset="utf-8">
     <title>Cisco DevNet: APIs, SDKs, Sandbox, and Community for
 Cisco Developers</title>

$ curl localhost:8080/time
The time is: Sun, 26 Jan 2020 20:11:22 CST
```

What’s left?

- Build the functions you need for your app
- Attach them to specific path
- Compile your code by into a Linux binary by running `go build <your_package>`
- Run your program when the container start using `start.sh`

Some final tips...

Developing the App (Frontend)

- You can create a script to package and publish the App for every incremental change you make
- But it's easier to create your App locally and publish for testing when it is finished
- APIC token must be managed locally and CORS (Cross Origin Resource Sharing) must be enabled
  - Access-Control-Allow-Origin:*
  - APIC certificate must be added to the trusted root certificates
- For larger project, use a streaming build system like gulp on node.js

Developing the App (Frontend)

```
//Set Variables
 var address = '1.1.1.1';
 var login = {
     aaaUser: {
         attributes: {
             name: 'admin',
             pwd: 'cisco123',
         }
     }
 };
 var url = 'https://' + address + '/api/mo/aaaLogin.json';

 //Get Token from aaaLogin
 function getToken(queryUrl, loginData){
     var token = '';
     $.ajax({
         url: queryUrl,
         method: 'POST',
         dataType: 'json',
         contentType: 'application/json',
         data: JSON.stringify(loginData),
         async: false,
         success: function(data){
             token = (data.imdata[0].aaaLogin.attributes.token)
         },
         error: function(error){
         console.log("===ERROR");
         console.log(error);
         },
     });
     return token;
 }
```

Developing with Docker (Backend)

- You're deploying from your local machine, not in the container, so:
  - Mount your local volumes to update the app
  - Avoid re-building your container every time you update your app
  - Any change in your local repo will be reflected in the container

Call to Actions

- App Center is a good way to start with ACI automation and delve into the object model
- Javascript is very popular to build frontend UI, getting knowledgeable about it is a good investment for your personal development
- Python is often used for automation, you can extend your skills by using Python in a different way, ie. building a backend application
- Or you can learn something new with Go!
- Think about something that would augment ACI experience in your environment (Some Ideas: Add historical changes in semi-structured DB, track snapshots, etc)
- Build it!

Other sources

- BRKACI-2945 Developing Your First ACI AppCenter Application
- BRKACI-1008 Cisco App Center for ACI
- DEVNET-1808 GoLang 101 for ITPros
- DEVNET-1338 How to build your first Network App by using Javascript and Cisco ACI Appcenter
- https://play.golang.org/ (Golang playground)

Complete your online session survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on ciscolive.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.com.

Continue your education

Demos in the Cisco Showcase

Walk-In Labs

Meet the Engineer 1:1 meetings

Related sessions

Thank you

CISCO

You make **possible**