

# Provisioning the IoT



Paul Bakker - @pbakker  
Sander Mak - @sander\_mak

Luminis Technologies

# Today's goals

1. Put IoT provisioning into context
2. Demo: modular provisioning
3. Apache ACE

# Provisioning spectrum

# Provisioning spectrum

servers

Cloud/SaaS:

- ▶ full control
- ▶ reliable network
- ▶ VM/Containers

# Provisioning spectrum

servers

mobile

Cloud/SaaS:

- ▶ full control
- ▶ reliable network
- ▶ VM/Containers

App stores:

- ▶ walled garden
- ▶ semi-reliable network
- ▶ full binaries

# Provisioning spectrum

IoT

?

# Provisioning?

'Just download latest binaries over FTP at system startup'



# Provisioning wishlist

Modularity

*Feature composition*

*Remotely toggle features*

Efficiency

*Bandwidth efficient*

*Avoid unnecessary updates*

Automation

*Manage many devices*

*Insight: what runs where?*

Security

*Not just any device*



# IoT != IoT

We are not talking about millions of devices:



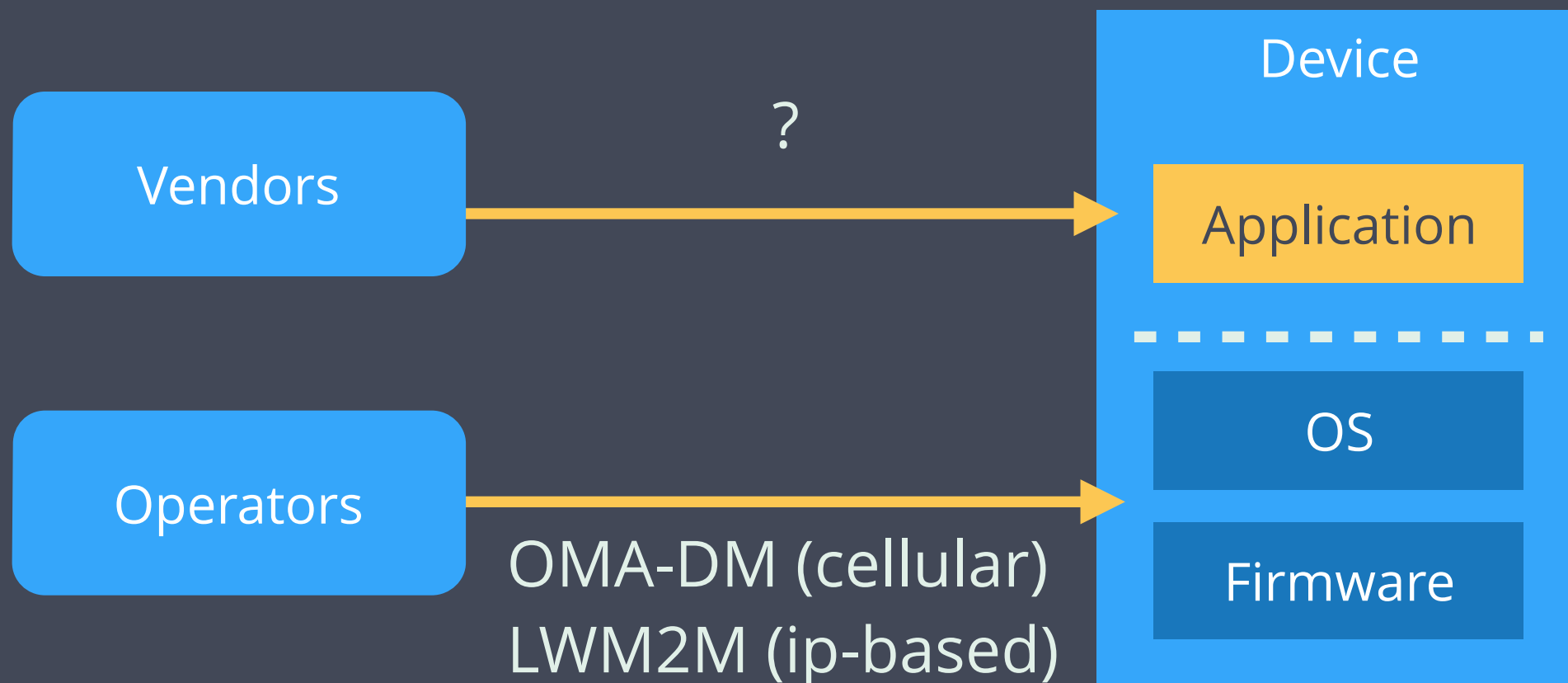
# IoT != IoT

We are not talking about millions of devices:



This talk is not about *device management*

# IoT Provisioning standards



Mostly about *device management*



## Software Update

There is a new version of your Tesla Model S software. Schedule installation, install now or close window to postpone.

12	00	
1	10	
<b>2</b>	<b>20</b>	<b>AM</b>
3	30	PM
4	40	

8 hr 47 min from now

SET FOR THIS TIME

INSTALL NOW



This update will take approximately 45 min

During the update process you will not be able to drive the vehicle or use the touchscreen, and your car alarm may be disarmed for a short duration. The car must be in PARK.





## Software Update


There is a new version of your Tesla Model S software. Schedule installation, install now or close window to postpone.

12	00	
1	10	
2	20	AM
3	30	PM
4	40	

8 hr 47 min from now

SET FOR THIS TIME

INSTALL NOW

 This update will take approximately 45 min. During the update process you will not be able to drive the vehicle or use the touchscreen, and your car alarm may be disarmed for a short duration. The car must be in PARK.

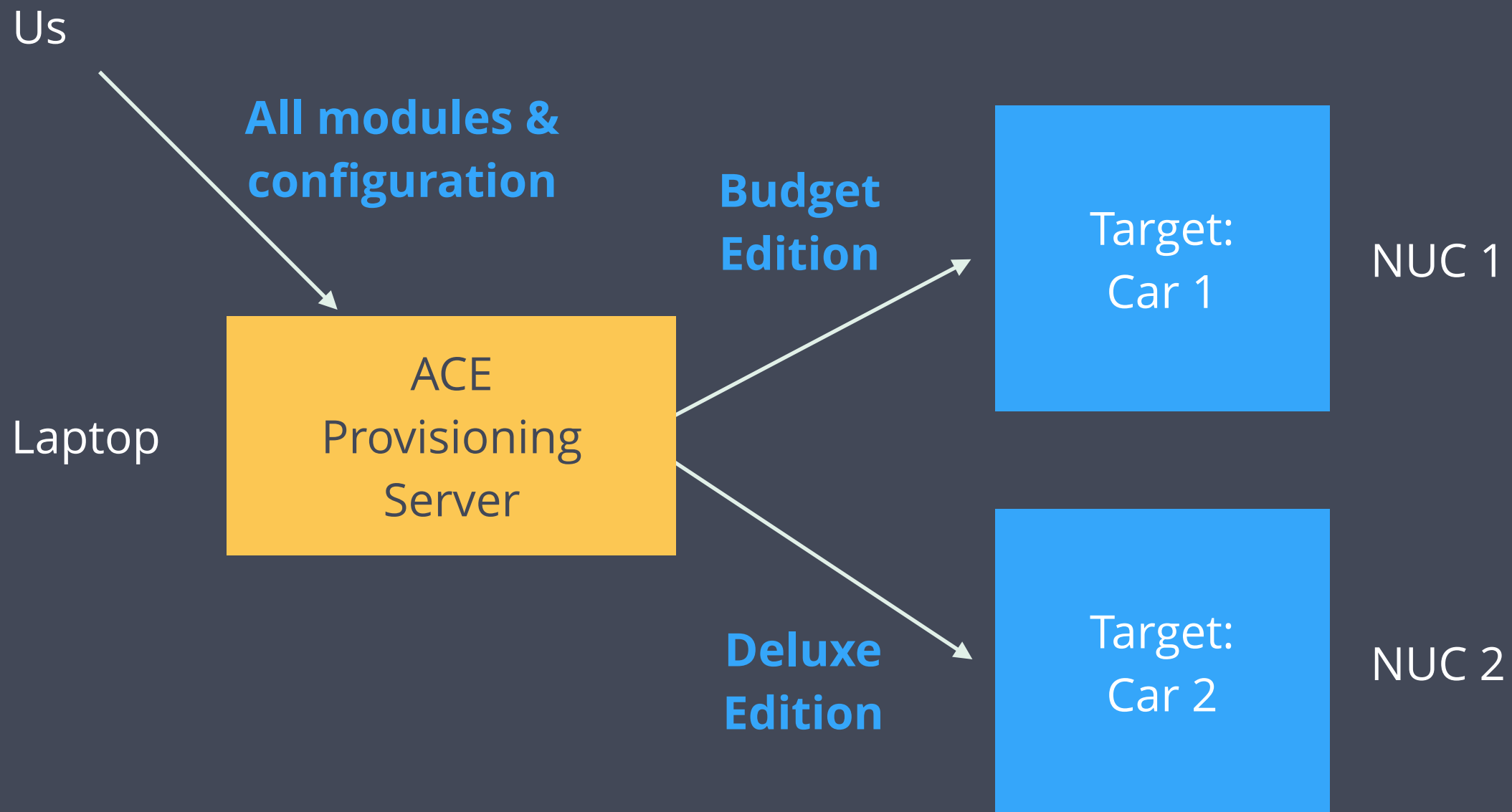
# Demo time!





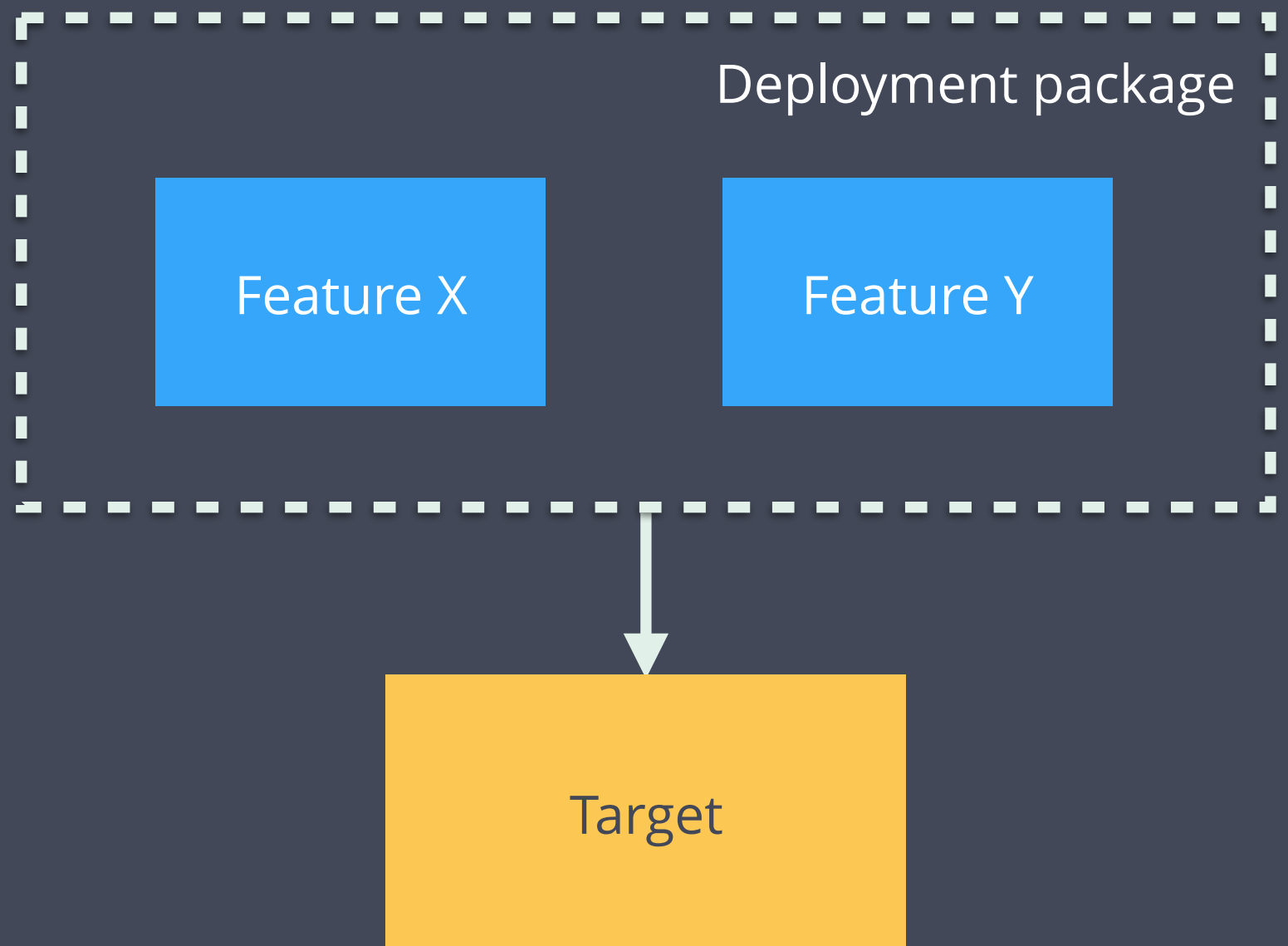
# Provisioning demo

*ACE Car Entertainment*



# Modular systems

*Monolithic deployment: installation*

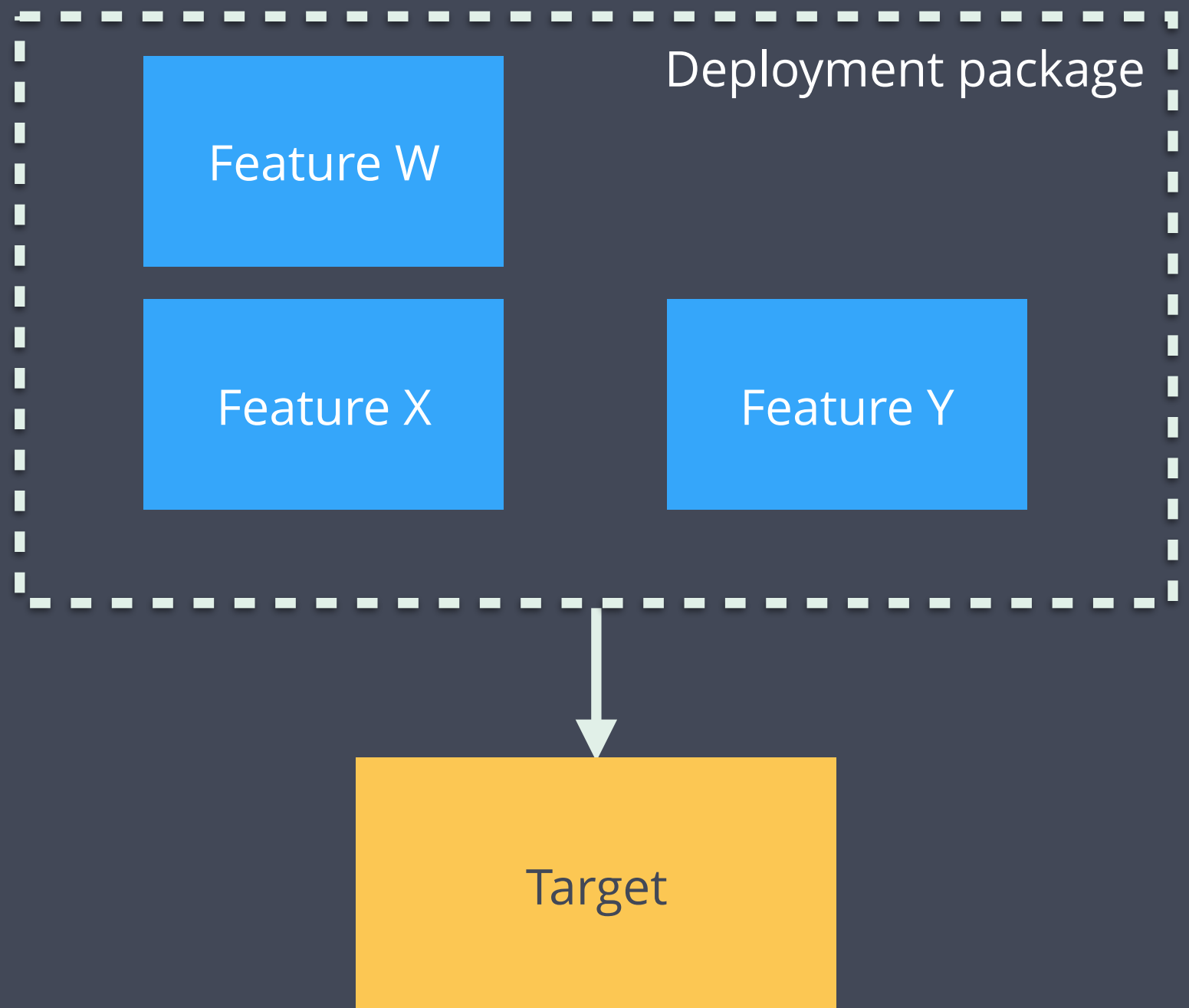


# Modular systems

*Monolithic deployment: update*



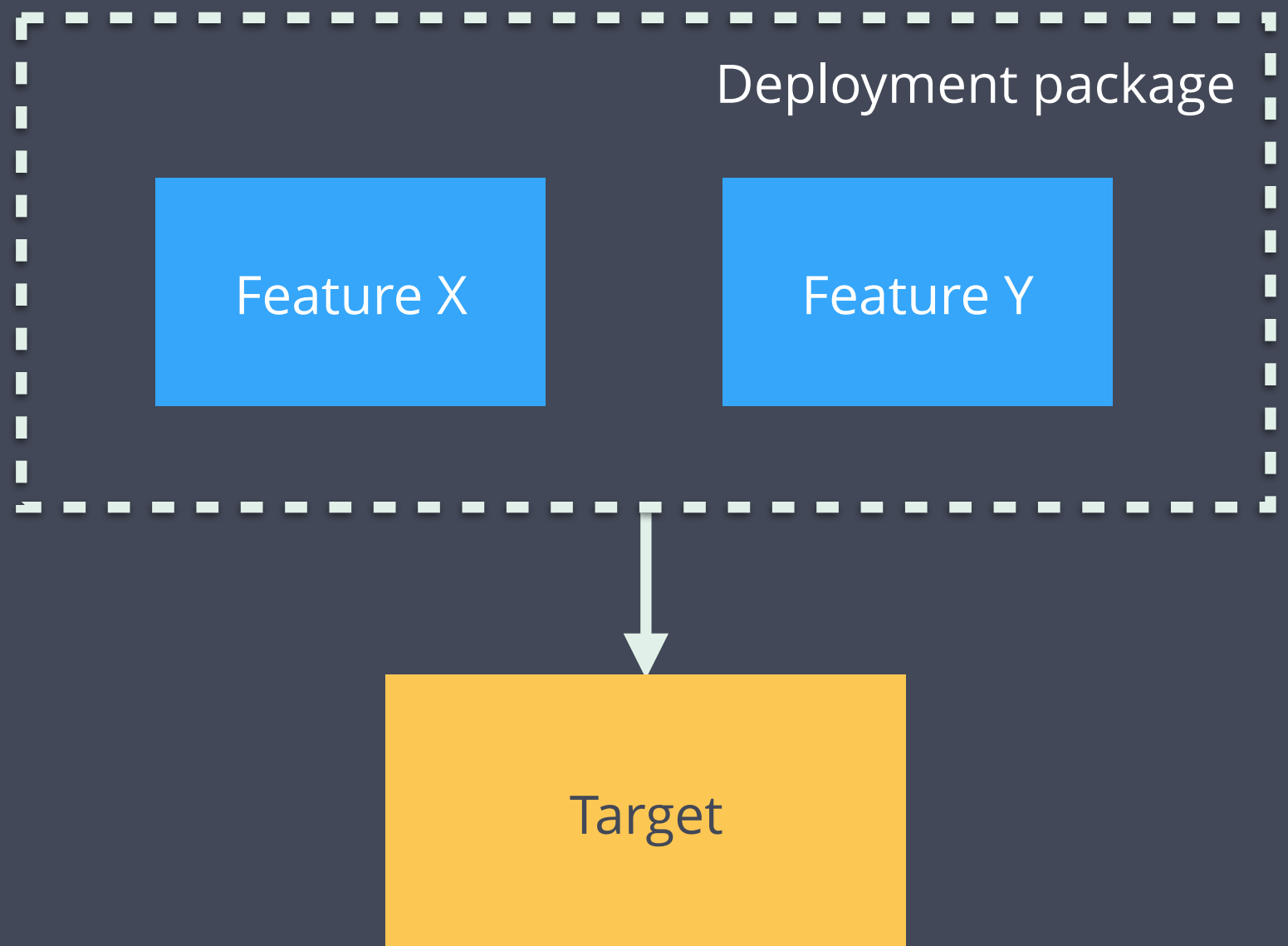
Limited connectivity  
Bandwidth inefficient





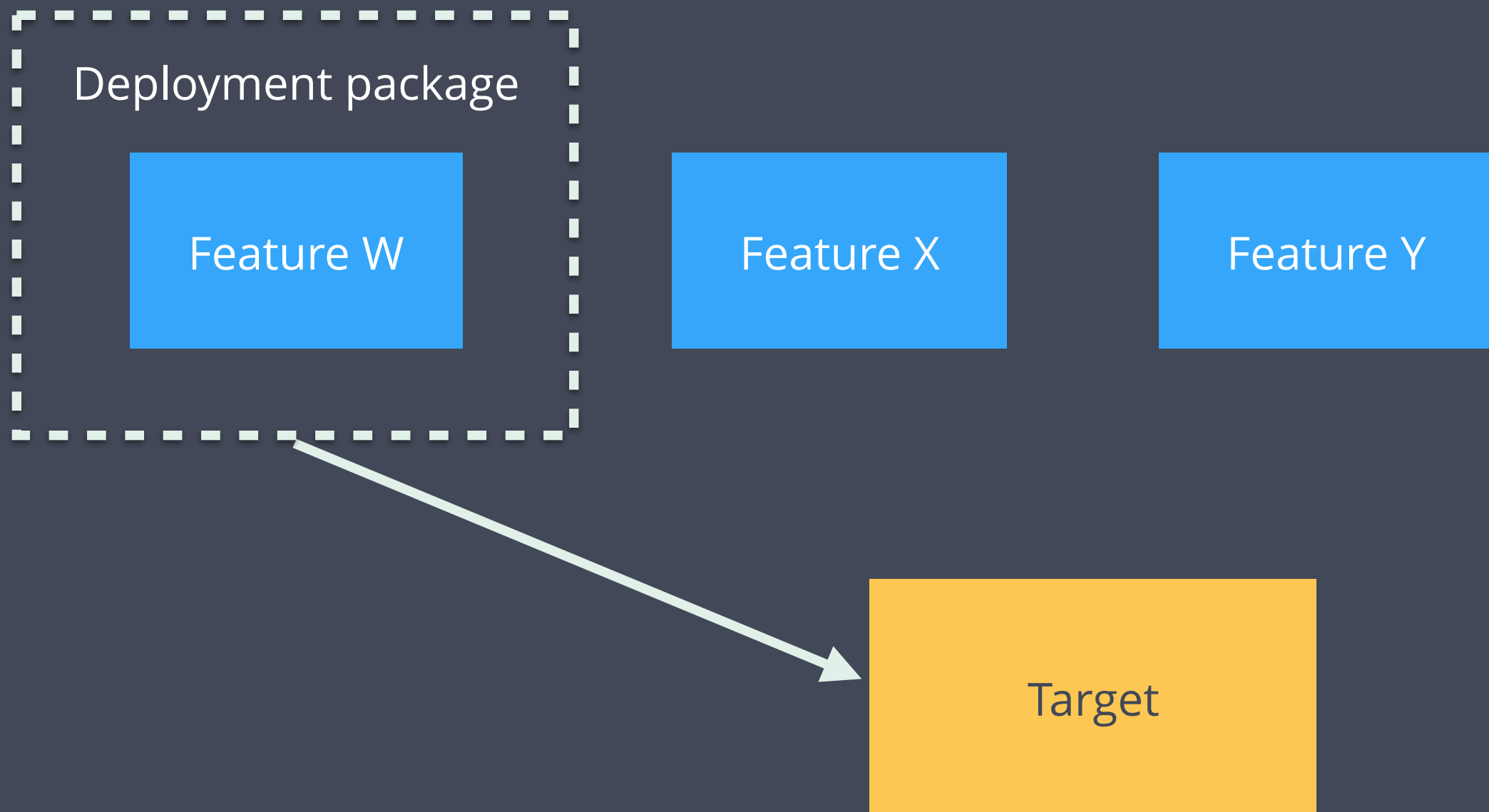
# Modular systems

*Modular deployment: installation*



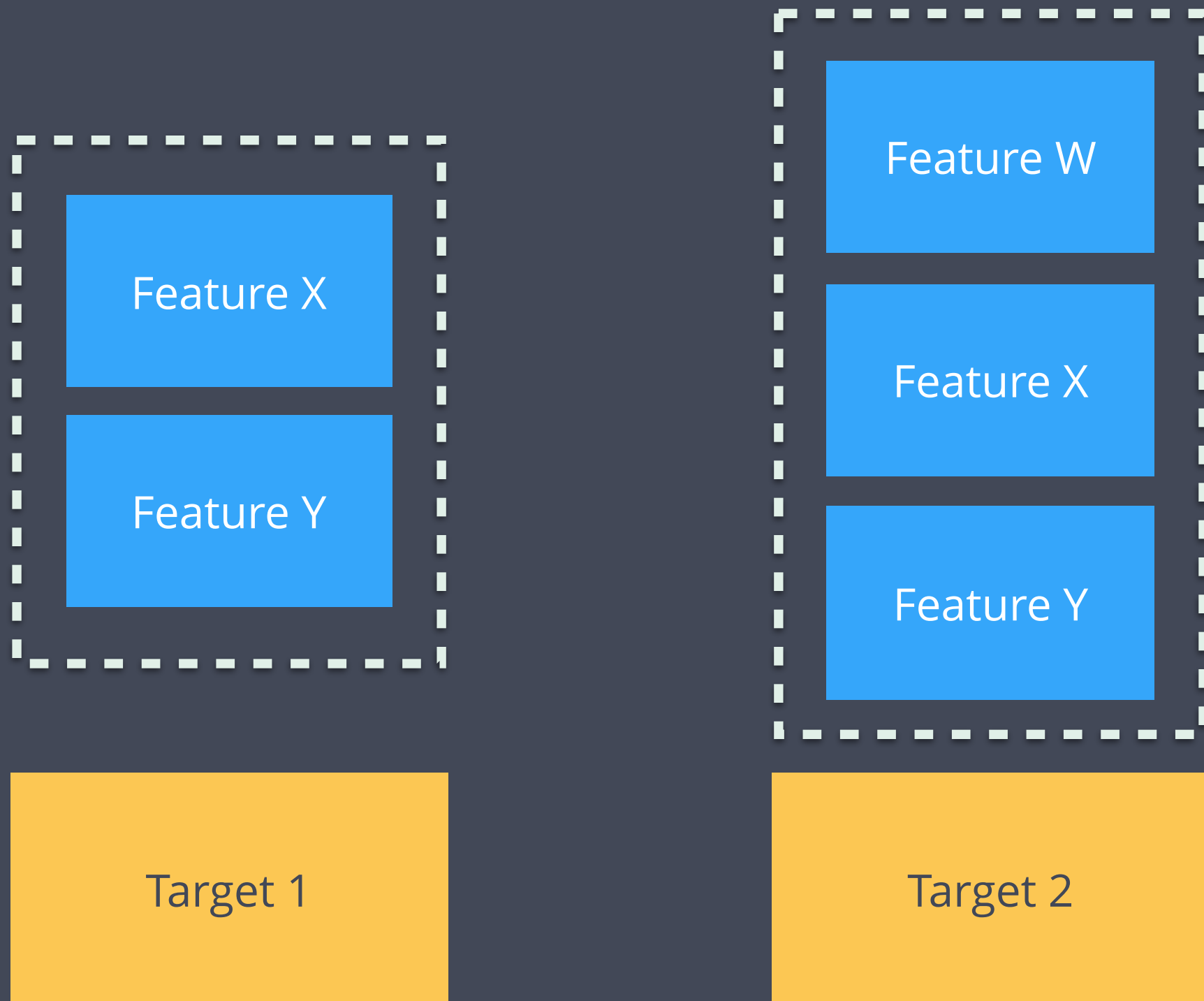
# Modular systems

*Modular deployment: update*



# Modular systems

*Differentiate with ease*



# Modular systems

*How?*



De facto standard for Java modularity

Small footprint

Dynamic service model



# OSGi

music  
v1.0.0

dashboard  
v1.0.0

phone  
v1.0.0

OSGi runtime (it's just Java!)

JVM



# OSGi

## Hot-swap bundles

music  
v1.0.0

dashboard  
v1.0.0

phone  
v2.0.0

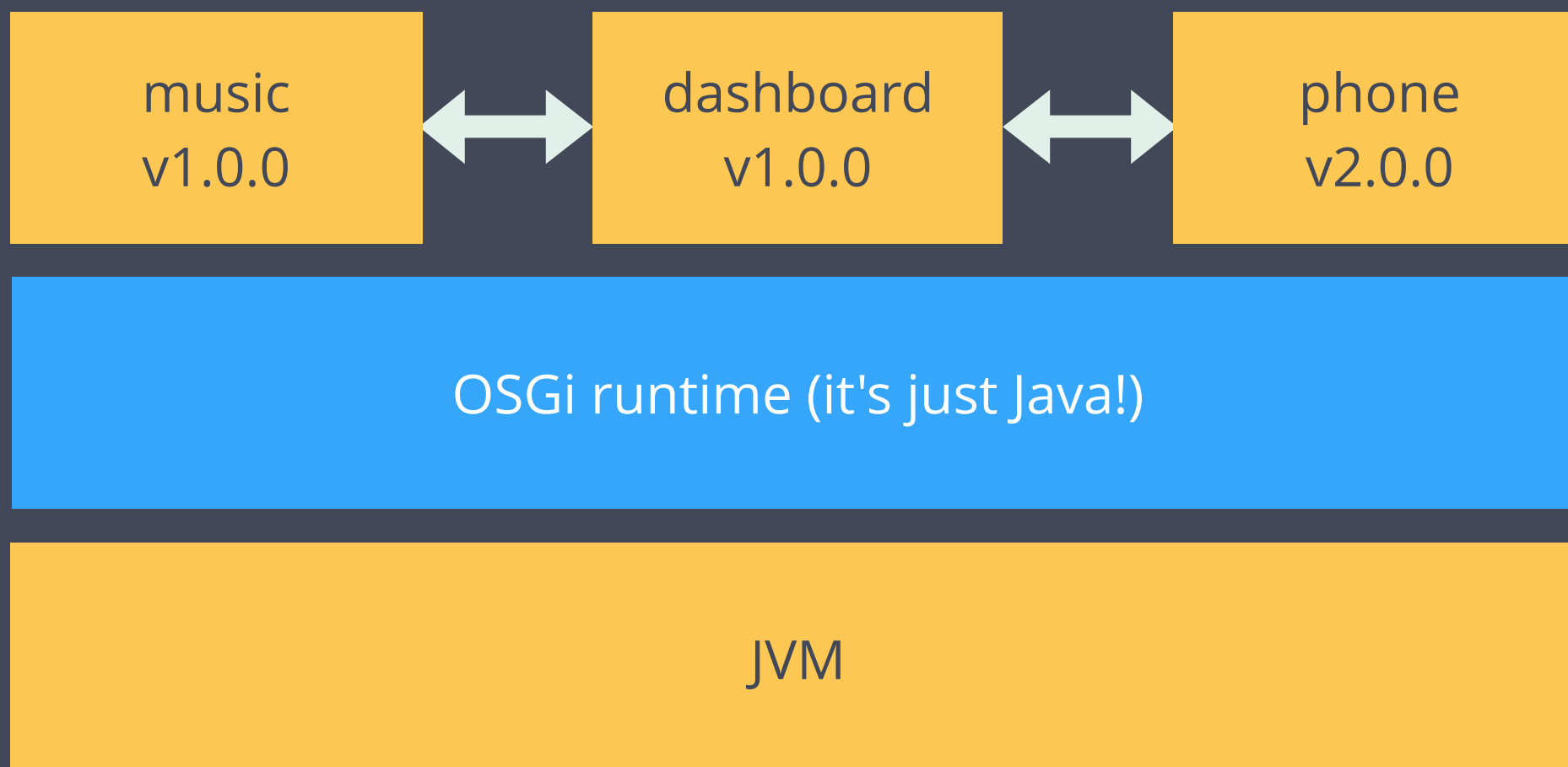
OSGi runtime (it's just Java!)

JVM



# OSGi

Dependencies explicit  
in bundle metadata





# OSGi

Service registry

App?

PhoneApp

music  
v1.0.0

dashboard  
v1.0.0

phone  
v2.0.0

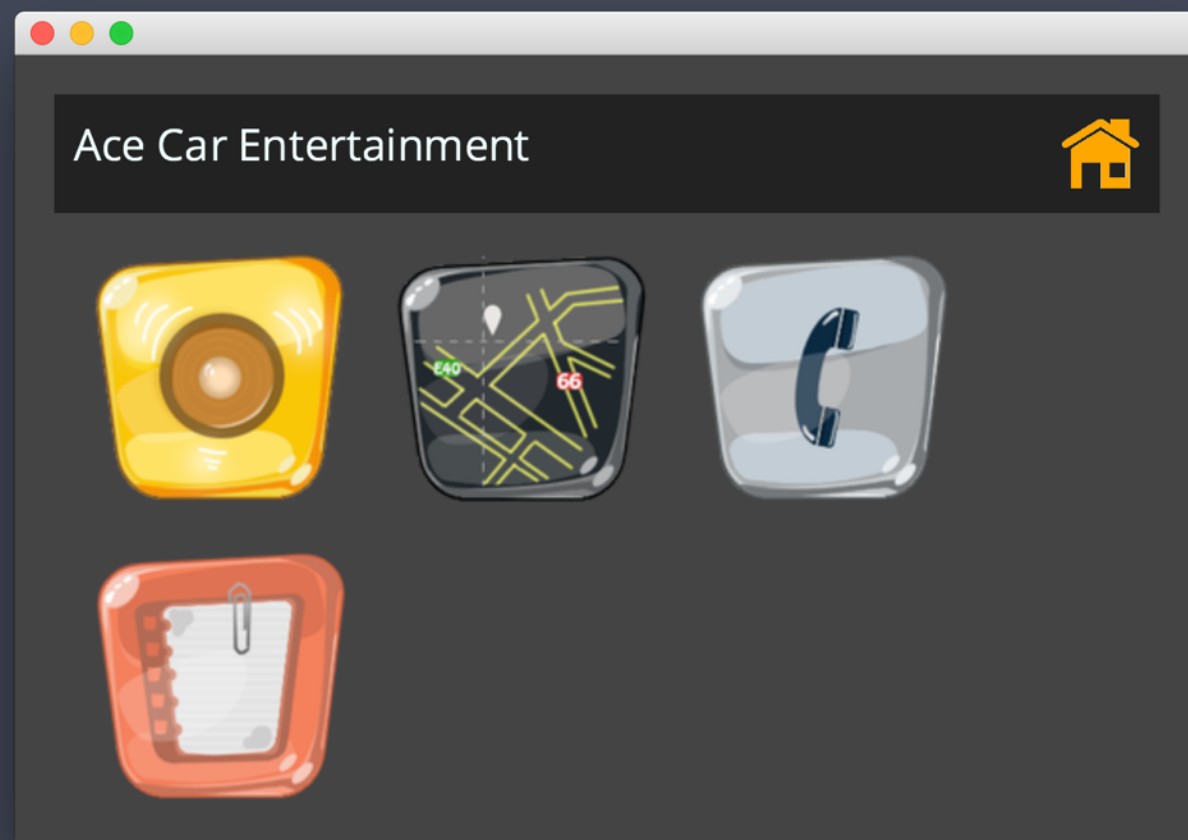
OSGi runtime (it's just Java!)

JVM



# Modular systems

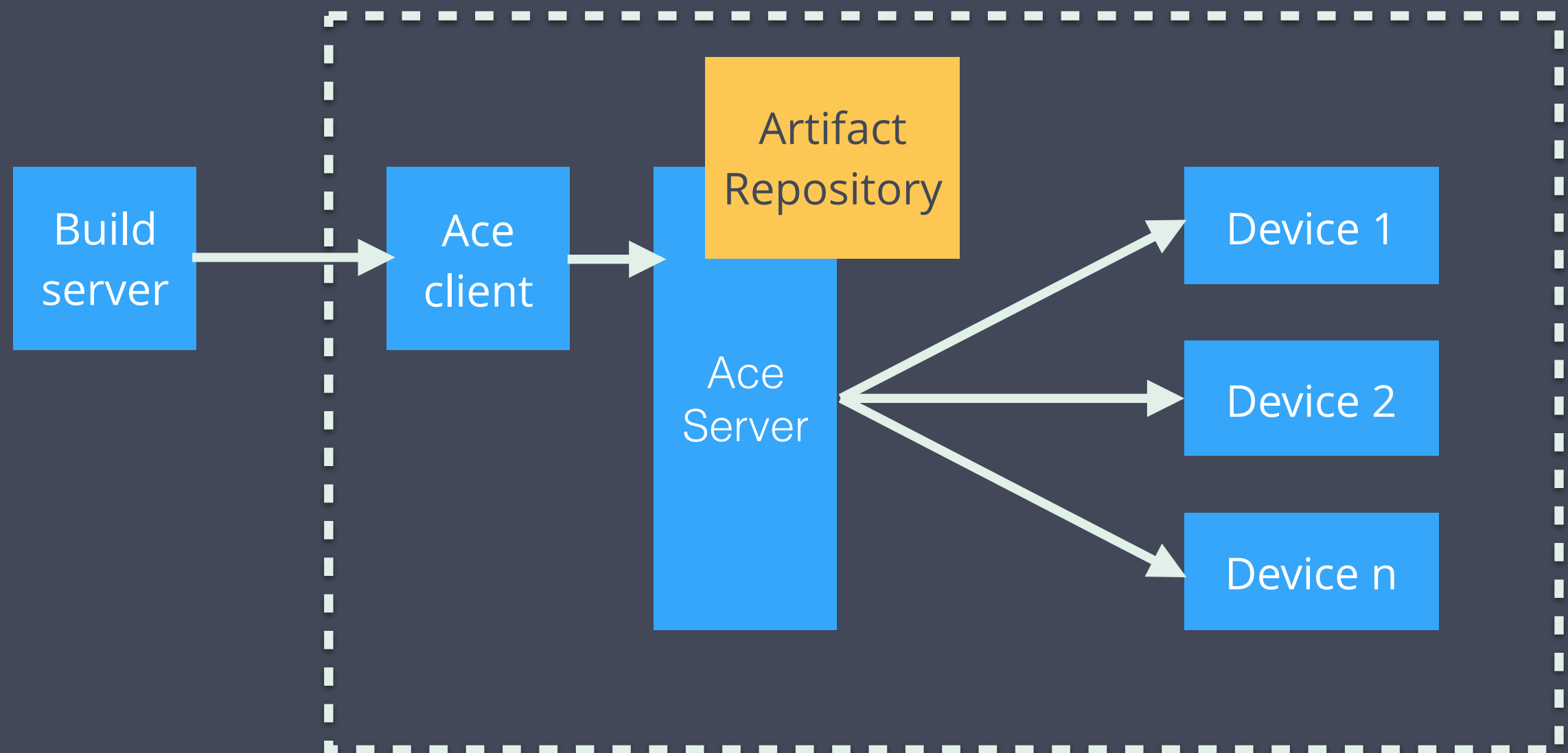
*Demo code*



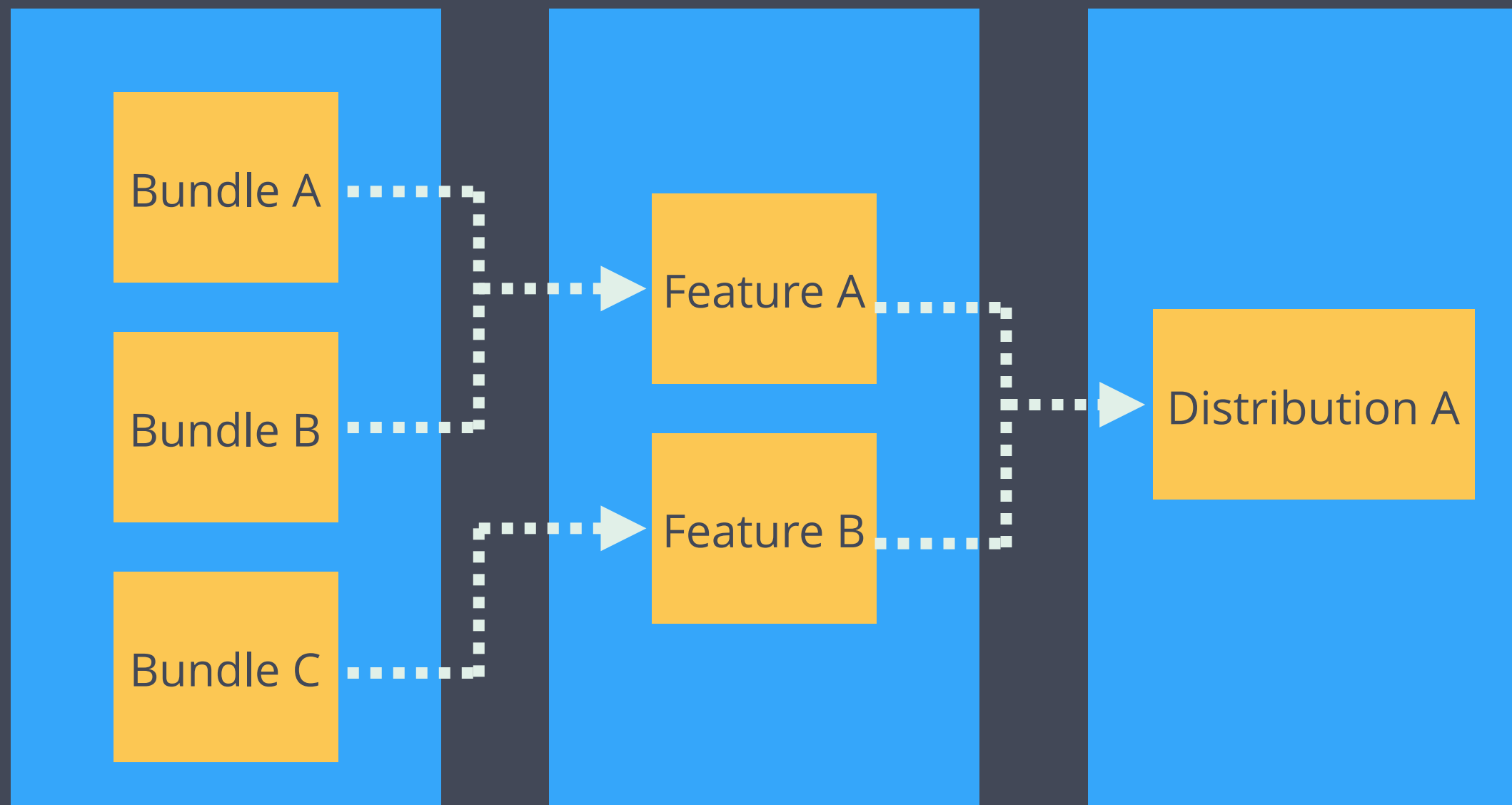
Code @ [bit.ly/carprov](https://bit.ly/carprov)



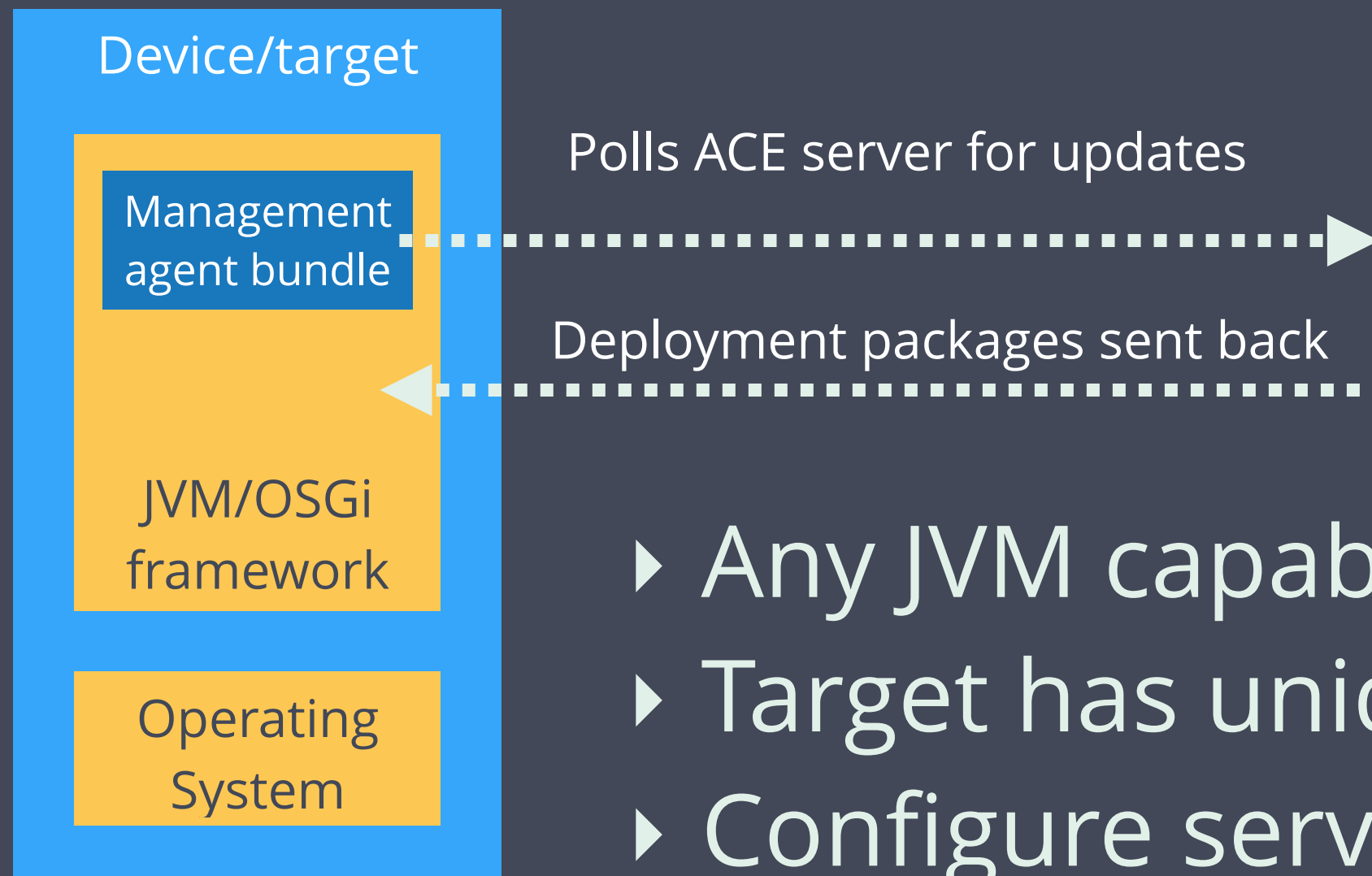
# Apache ACE Architecture



# Apache ACE Model



# Apache ACE Targets



- ▶ Any JVM capable device
- ▶ Target has unique id
- ▶ Configure server location

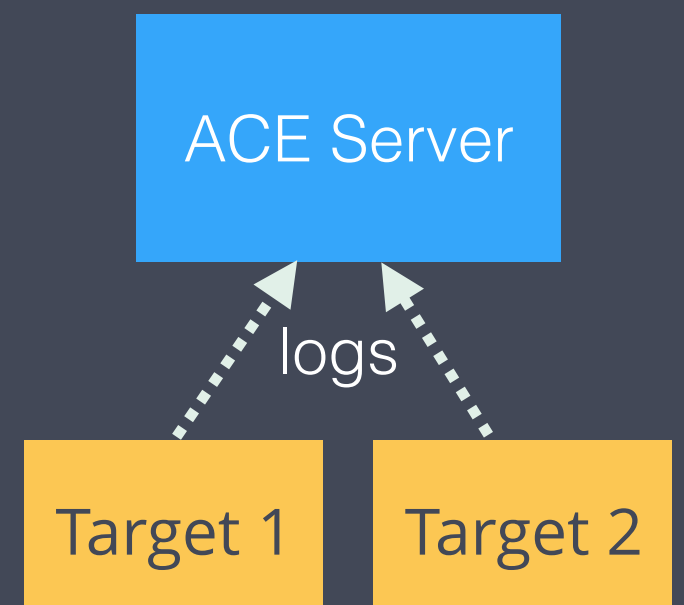


## Security:

- ▶ Explicit target registration
- ▶ HTTP auth or SSL Client Certificates

## Process:

- ▶ Manual update approval
- ▶ Properties per feat/dist/target
- ▶ Audit logs





REST

GET /work/{id}/feature

Java API

workspaceMgr.cw().lf()

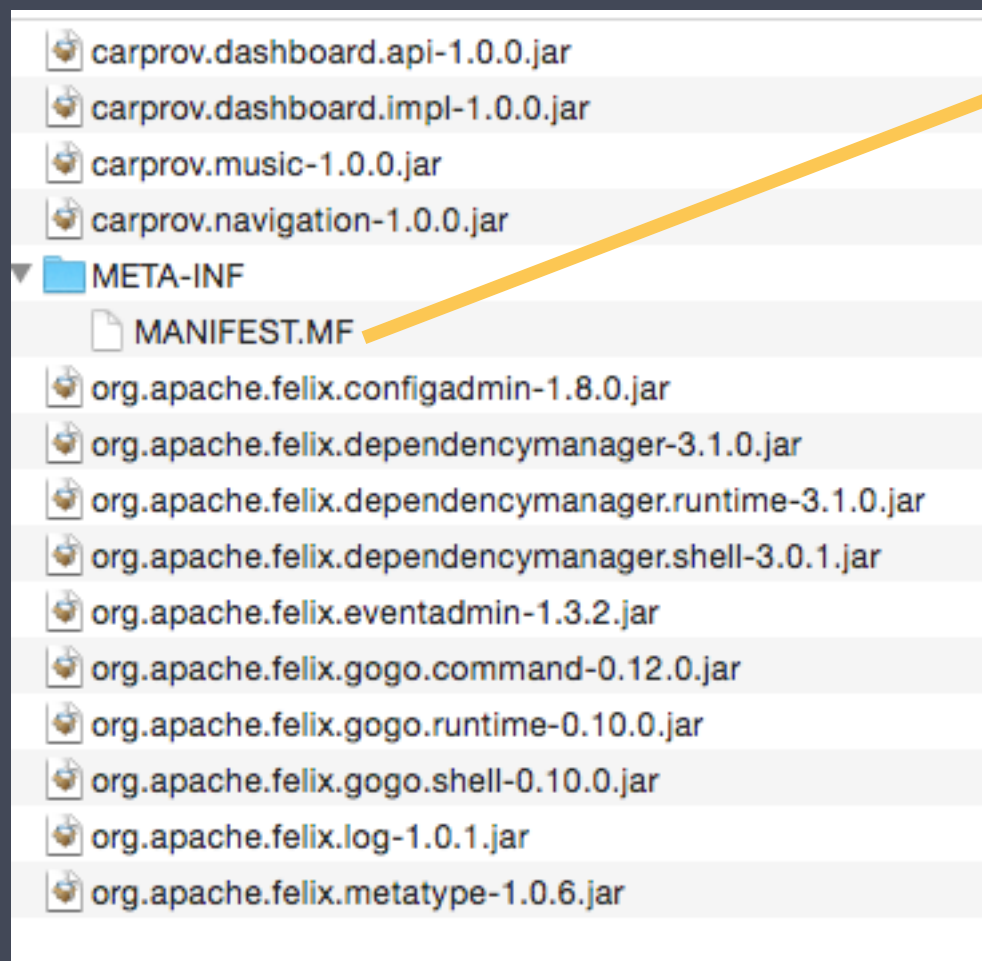
Gogo script

scriptable OSGi shell, calls (low-level) Java API

# Deployment packages

## *Structure*

GET /deployment/car1/versions/3.0.0



### META-INF/MANIFEST.MF

Manifest-Version: 1.0

DeploymentPackage-SymbolicName: car1

DeploymentPackage-Version: 3.0.0

Name: carprov.dashboard.impl-1.0.0.jar

Bundle-SymbolicName: carprov.dashboard.impl

Bundle-Version: 1.0.0

Name: org.apache.felix.dependencymanager-3.1.0.jar

Bundle-SymbolicName:

org.apache.felix.dependencymanager

Bundle-Version: 3.1.0

...



# Deployment packages

*Installation on target*

- ▶ Transactional: retries and rollback
- ▶ Installation status in audit log
- ▶ Unreliable networks:
  - ▶ Download instead of stream
  - ▶ Resumable downloads
  - ▶ Custom update strategies

# ACE Extensibility

- ▶ Built on modular OSGi architecture
- ▶ Different repository implementations
- ▶ Custom update strategies on targets
- ▶ Multiple topologies (e.g. relay server)
- ▶ ResourceProcessors for new artifact types

# ACE Extensibility

## *ResourceProcessor*

- ▶ Recognizes your artifact type
- ▶ Handles installation on target
- ▶ Upload ResourceProcessor to ACE

# ACE Extensibility

## *Configuration ResourceProcessor*

- ▶ Handles XML (MetaType) configs
- ▶ Placeholders replaced with target tags

On server:

```
...  
<Value>${context.mySpecificProperty}</Value>  
...
```

On target:

```
...  
<Value>mySpecificValue</Value>  
...
```

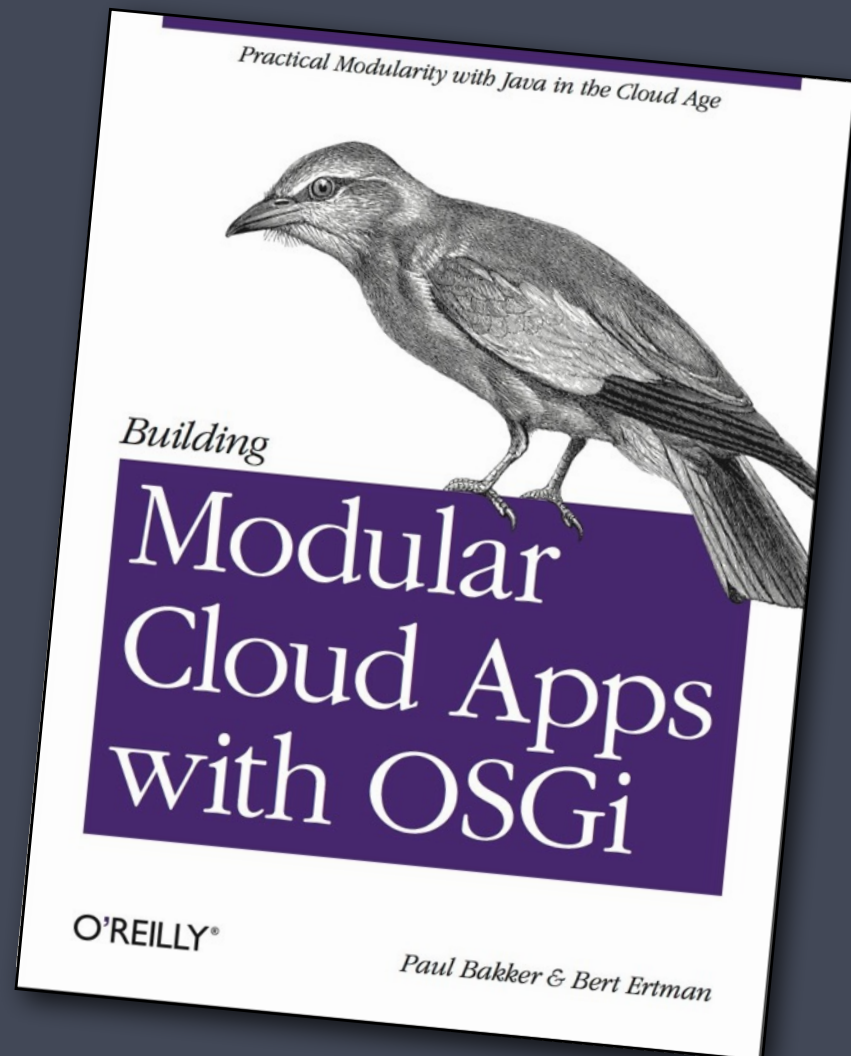
# In summary

Apache ACE simplifies IoT deployments

Modular applications rock!

Modular deployment is worth your while

# Thank you!



Code @  
[bit.ly/carprov](http://bit.ly/carprov)

Paul Bakker - @pbakker

Sander Mak - @sander\_mak

