what’s new with CF Services?

async operations, service keys, and arbitrary parameters

Michael Maximilien and Beijing team (IBM)
David Sabeti (Pivotal)

v1.0.0
May 11th, 2015

agenda

CF services - architecture (brief history)

what’s new?

(1) async service operations
(2) service keys
(3) arbitrary parameters

demo

next steps

CF services - architecture - problem

problem

enable developers to **discover** and **integrate third-party software** (services) with their CF applications **through the platform**

CF services - architecture - problem

discover third-party software:

```
> cf marketplace

p-mysql    100mb, 1gb   MySQL databases on demand
p-riakcs   developer    An S3-compatible object store based on Riak CS
```

CF services - architecture - problem

integrate:

```
> cf create-service myService somePlan myDB
Creating service instance myDB in org myOrg / space mySpace as myUser...
OK

> cf bind-service myApp myDB
Binding service myDB to app myApp in org myOrg / space mySpace as myUser...
OK

> cf restage myApp
...

// Your app can now use your service
```

CF services - architecture - problem

through the platform

CF services - architecture - other goals

other goals

deployment and maintenance are **opaque** to developer

CF services - architecture - other goals

other goals

don’t be prescriptive about what a service is

CF services - architecture - other goals

other goals

external vendors can bring their service to CF

CF services - architecture - other goals

other goals

added, discovered, & integrated dynamically

CF services - architecture

solution

the cf services model (service broker architecture)

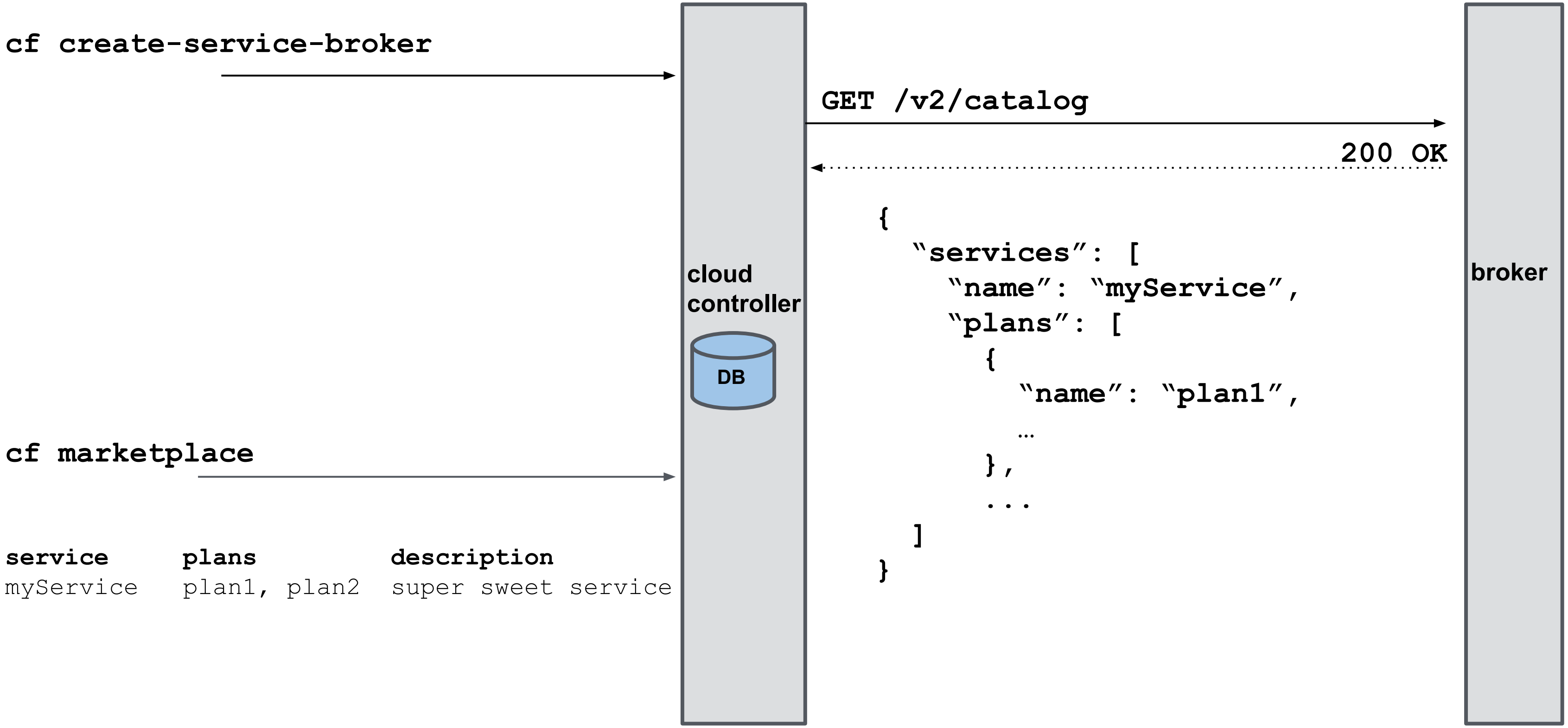
CF services - architecture

service brokers

translation layer

small HTTP interface

CF services - architecture



```
cf create-service-broker
```

```
GET /v2/catalog
```

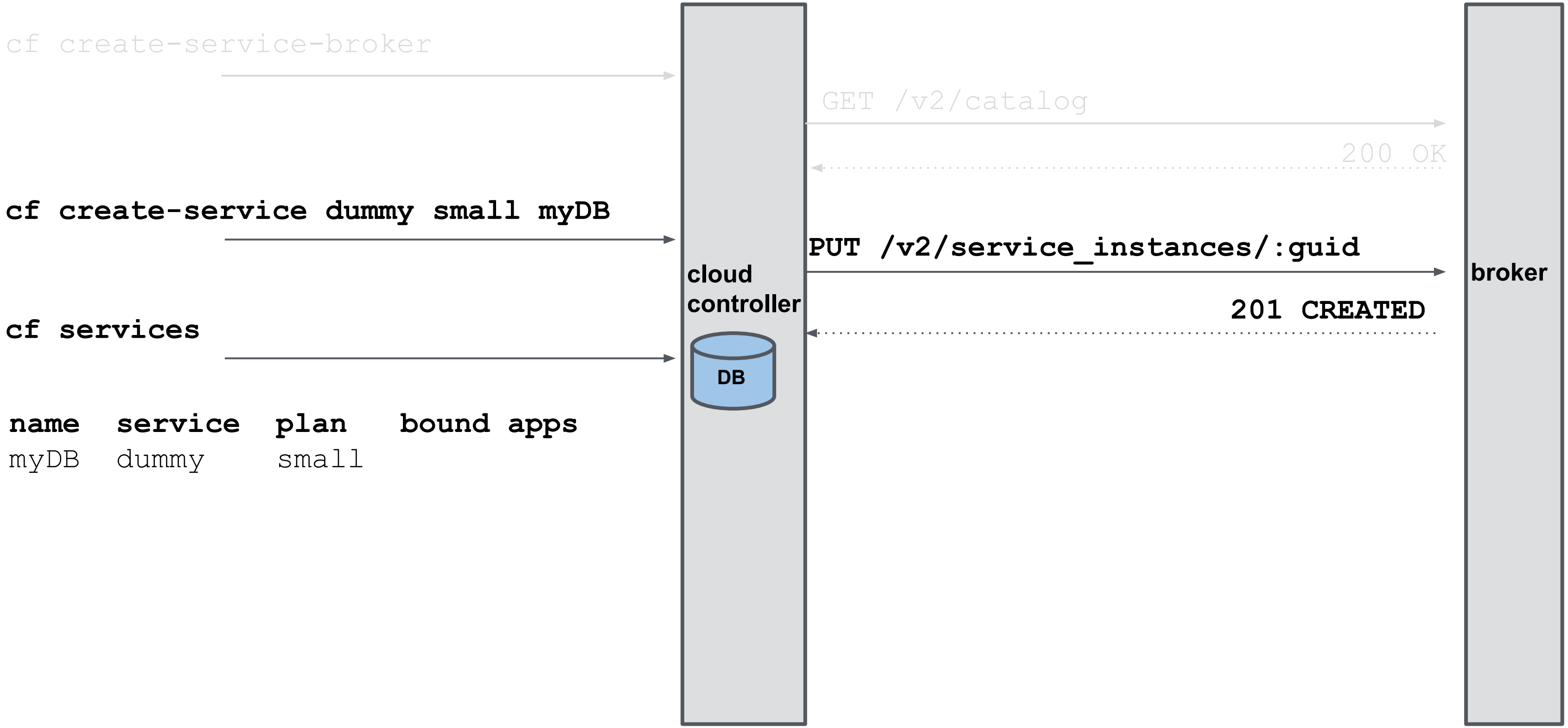
```
200 OK
```

```
{
  "services": [
    "name": "myService",
    "plans": [
      {
        "name": "plan1",
        …
      },
      ...
  ]
}
```

```
cf marketplace
```

```
service     plans          description
myService   plan1, plan2   super sweet service
```

CF services - architecture



```
cf create-service-broker
```

```
GET /v2/catalog
200 OK
```

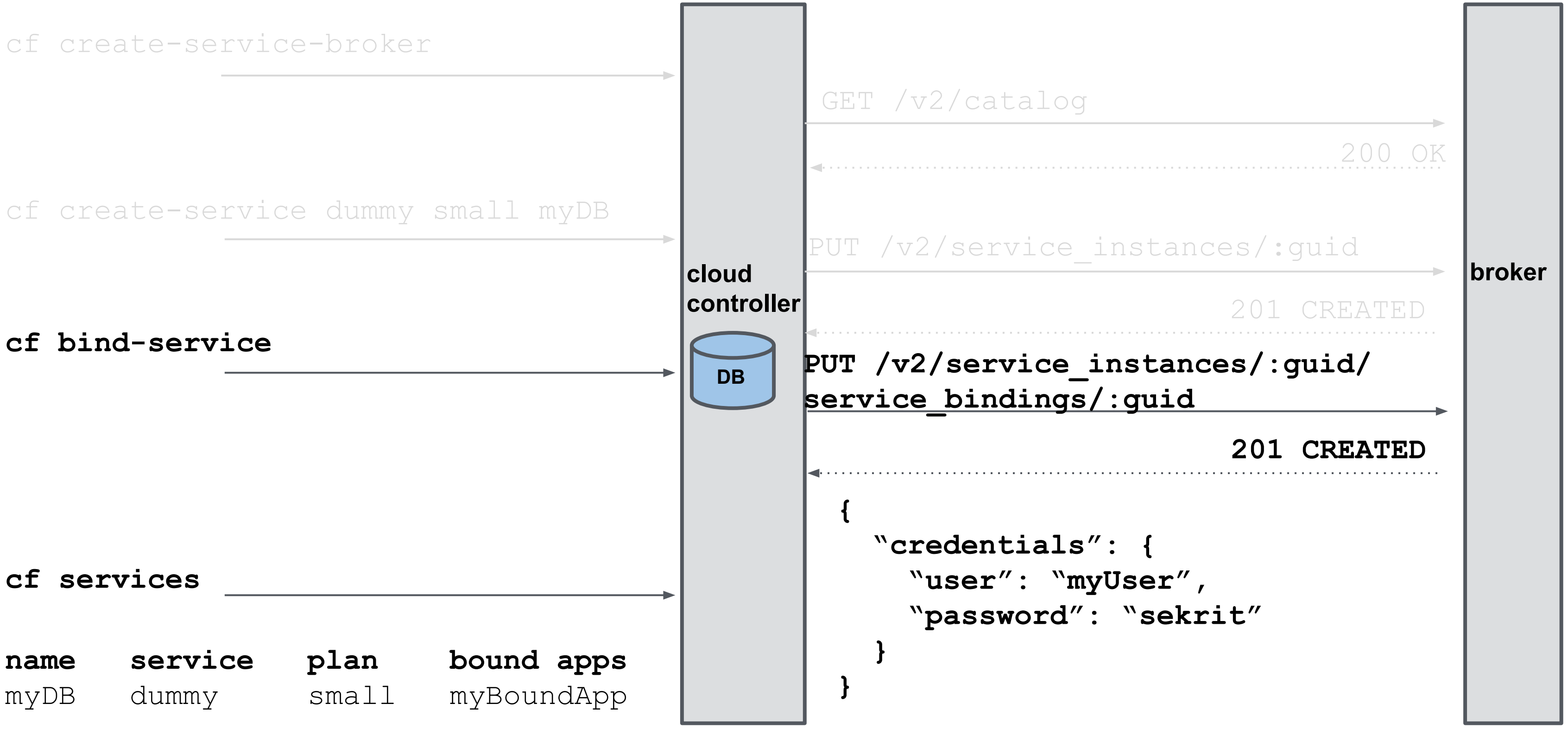
```
cf create-service dummy small myDB
```

```
PUT /v2/service_instances/:guid
201 CREATED
```

```
cf services
```

```
name   service   plan   bound apps
myDB   dummy     small
```

CF services - architecture



```
cf create-service-broker
```

```
GET /v2/catalog
```

```
200 OK
```

```
cf create-service dummy small myDB
```

```
PUT /v2/service_instances/:guid
```

```
201 CREATED
```

```
cf bind-service
```

```
PUT /v2/service_instances/:guid/
service_bindings/:guid
```

```
201 CREATED
```

```
{
  "credentials": {
    "user": "myUser",
    "password": "sekrit"
  }
}
```

```
cf services
```

```
name    service    plan     bound apps
myDB    dummy      small    myBoundApp
```

CF services - what’s new

so what’s new?

asynchronous operations

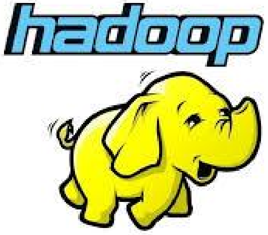
problem

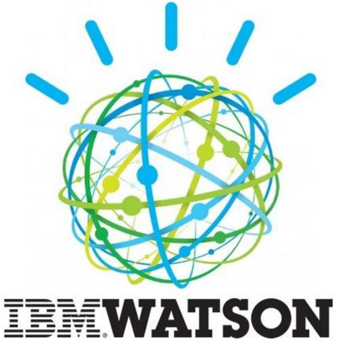
how does the existing API support long running operations?

asynchronous operations

use cases

long running provisioning and de-provisioning





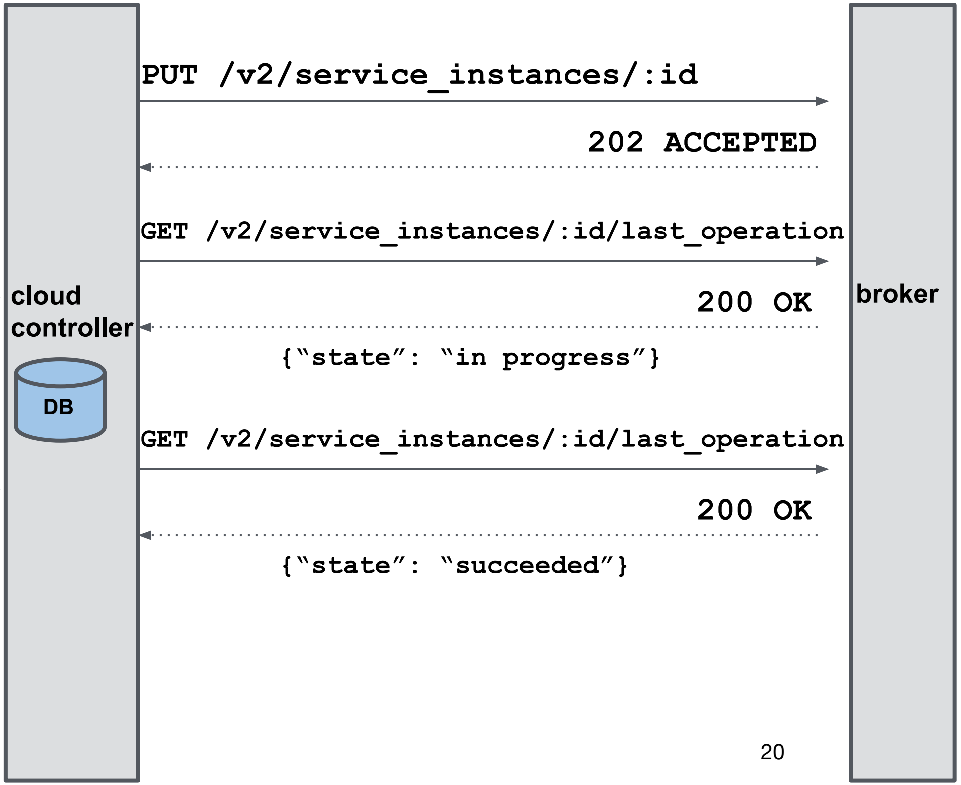
asynchronous operations

solution

future object: `last_operation` with state

- clients must poll to find service instance operation state
- tighter CF <-> broker interaction

asynchronous operations



cf create-service

PUT /v2/service_instances/:id

202 ACCEPTED

cf services

GET /v2/service_instances/:id/last_operation

```
name   service  plan   bound apps   last_operation
myDB   dummy    small               create in progress
```

cloud controller

DB

200 OK

{“state”: “in progress”}

broker

cf services

GET /v2/service_instances/:id/last_operation

```
name   service  plan   bound apps   last_operation
myDB   dummy    small               create in progress
```

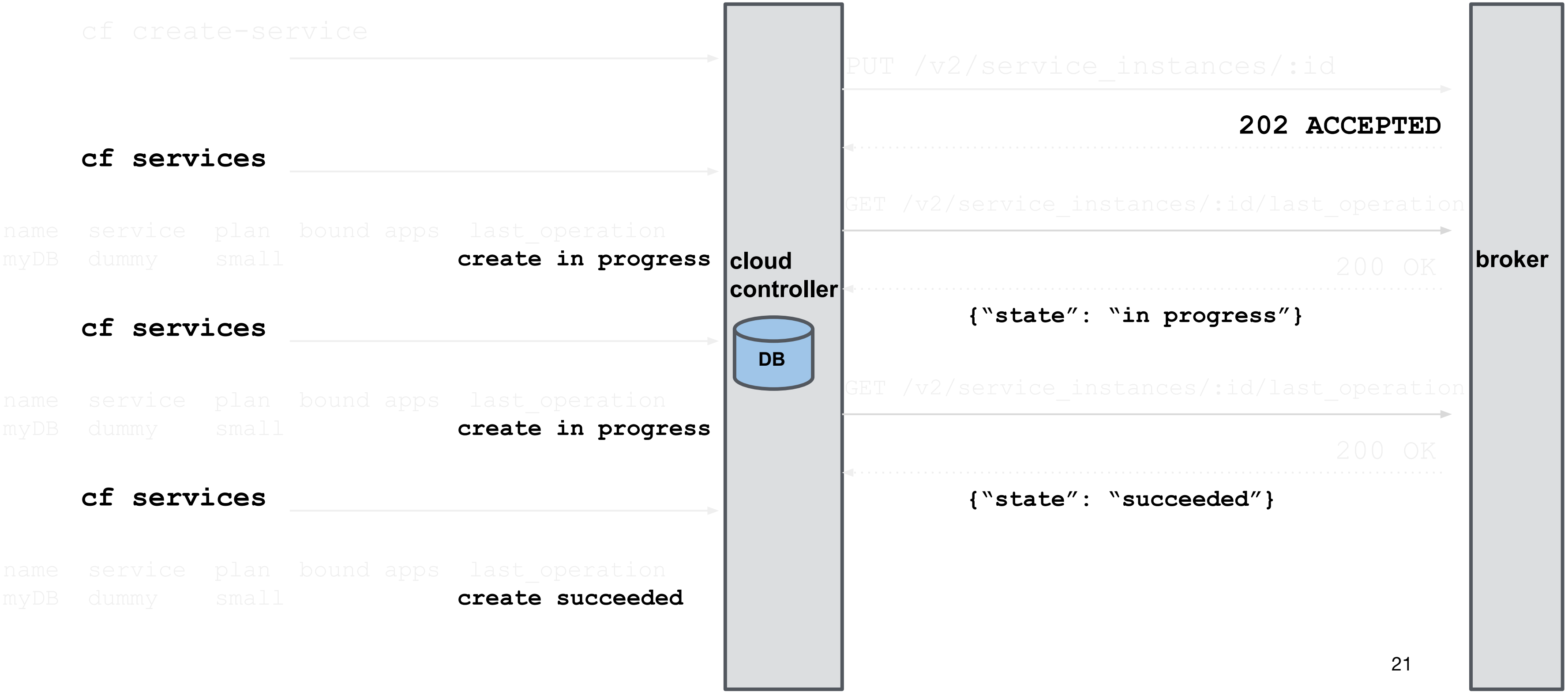
200 OK

{“state”: “succeeded”}

cf services

```
name   service  plan   bound apps   last_operation
myDB   dummy    small               create succeeded
```

asynchronous operations



cloud controller

DB

broker

```
cf create-service
```

PUT /v2/service_instances/:id

202 ACCEPTED

```
cf services
```

GET /v2/service_instances/:id/last_operation

200 OK

{"state": "in progress"}

```
name   service   plan   bound apps   last_operation
myDB   dummy     small               create in progress
```

```
cf services
```

GET /v2/service_instances/:id/last_operation

200 OK

{"state": "succeeded"}

```
name   service   plan   bound apps   last_operation
myDB   dummy     small               create in progress
```

```
cf services
```

```
name   service   plan   bound apps   last_operation
myDB   dummy     small               create succeeded
```

services keys

problem

CF services must be **bound** to an application to **obtain credentials** via platform

services keys

use cases: using services outside of CF

- **multiple** credentials per service instance
- **privileged** credentials for admins
- **read-only** credentials for some apps
- easier **accounting** for service usage

services keys

solution: creation of new endpoint

- `POST /v2/`**`service_keys`**`/service_instance_guid`
- `GET /v2/`**`service_keys`**`/:guid`
- `DELETE /v2/`**`service_keys`**`/:guid`

… and the broker API requires no changes because we can use binding endpoints

arbitrary service parameters

problem

current CF Service APIs **limit** what can be passed to a service broker during service lifecycle operations

arbitrary service parameters

use cases

Optional service configuration

- ex. Number of nodes in a cluster

Instance-specific configuration

- service plans assume all instances need same config
- removing dashboard work-around
- ex. SSL certificates as a service

arbitrary service parameters

solution

service lifecycle operations take **additional JSON parameter**

arbitrary service parameters

solution

```
$cf create-service my-service somePlan myDB
```

then out-of-band configuration via UI

is now

```
$cf create-service my-service somePlan myDB -p “{“sharding”: true,
“size”: “large”, ...}”
```

demo

Async: https://www.youtube.com/watch?v=Ij5KSKrAq9Q
Keys: https://www.youtube.com/watch?v=V5uzLcPQPmo
Params: https://www.youtube.com/watch?v=Qc3bZIjGscs

feedback

need feedback from community

- real-world usage will drive future improvements
- where else can CF services improve?
- contact Shannon Coen (scoen@pivotal.io)

team

Raina Masand
(incoming anchor)

Whitney Schaefer

Yotam Kadishay

Shannon Coen
(Product Manager)

Glen Tona

Jeff Hui

Greg Cobb

thank you

credit: http://knowyourmeme.com/photos/522333-language

backup

backup