Automatic Deployment and Execution of Applications using Cloud Services

CSIC
Consejo Superior de Investigaciones Científicas

CESGA

e-infrastructure

Main objective

Develop a contextualization service that aids scientific communities to execute their computing workload by automating the deployment of scientific software on virtual machines using the interfaces and standards within EGI's Federated Cloud

Why contextualize?

- Configuration not known until instantiation
  - e.g. location of data / servers
- Private Information
  - e.g. host certs
- Software that changes frequently / under development
  - Not practical to create a new VM image every time

Support from the Testbed

- Contextualization requires passing some data to the VMs on instantiation
- Initially OCCI API lacked such feature
  - New mixin proposed and implemented in rOCCI server and OCCI-OS
    - org.openstack.user_data
    - base64 enconded data
  - Sites now supporting it!

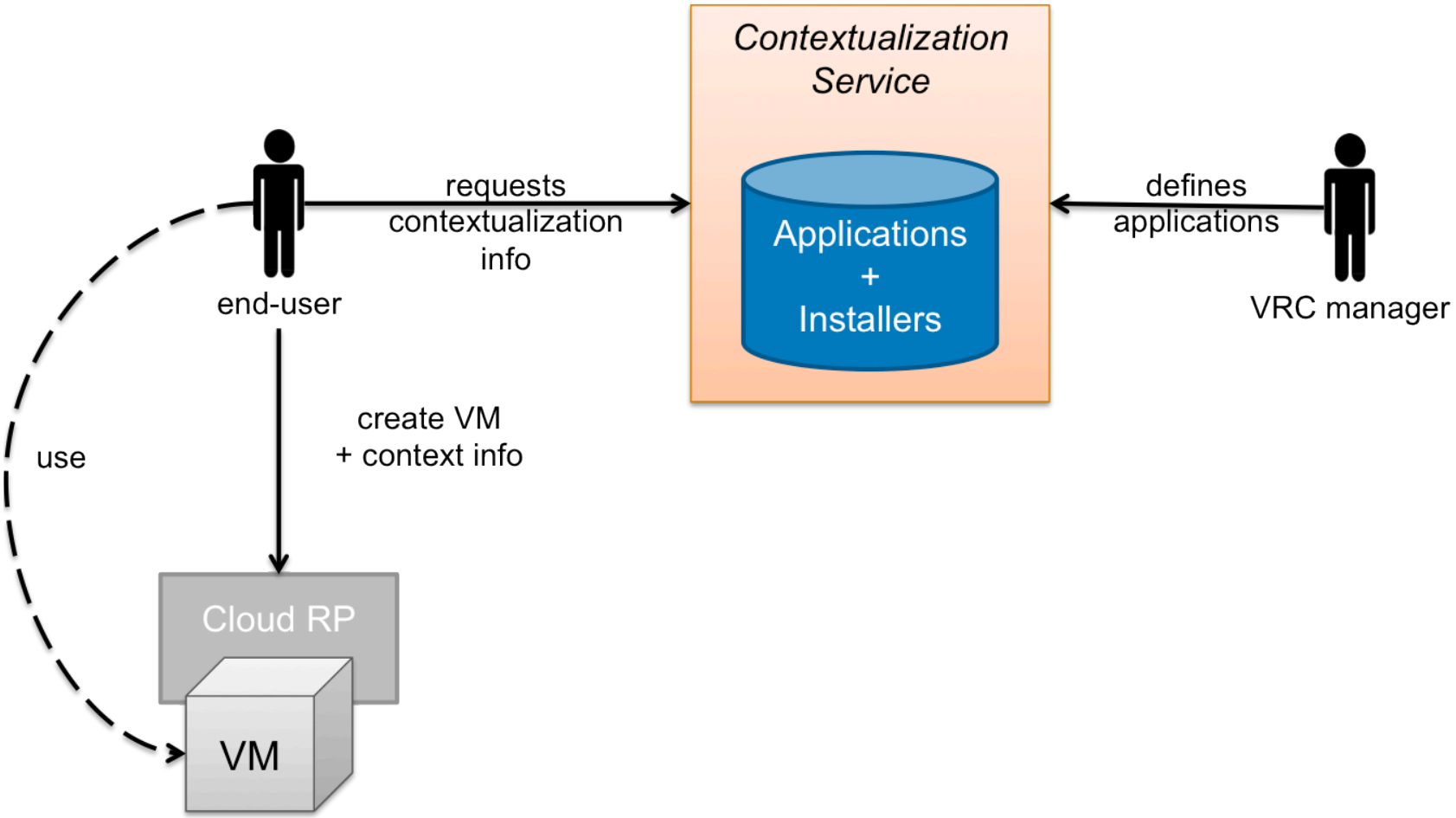
Using the data at the VM

- Each RP has different mechanisms to provide the data:
  - metadata server, iso filesystem or file injection
- Cloud-init abstracts these mechanisms and defines a format for the data
  - configures: network, users, services, repositories, CAs, ssh-keys, filesystems, …
  - installs packages
  - executes arbitrary commands
  - executes user provided scripts

Cloud-init support

- OpenStack uses EC2 metadata server
  - cloud-init already supports it
- OpenNebula includes context info in a `context.sh` file in an iso filesystem
  - New branch of cloud-init supports fetching OpenNebula contextualization info
  - https://code.launchpad.net/~vlastimil-holer/cloud-init/opennebula
  - Tested successfully with ubuntu VM (see demo later on)
- Testing now on RH based distros

Contextualization Service

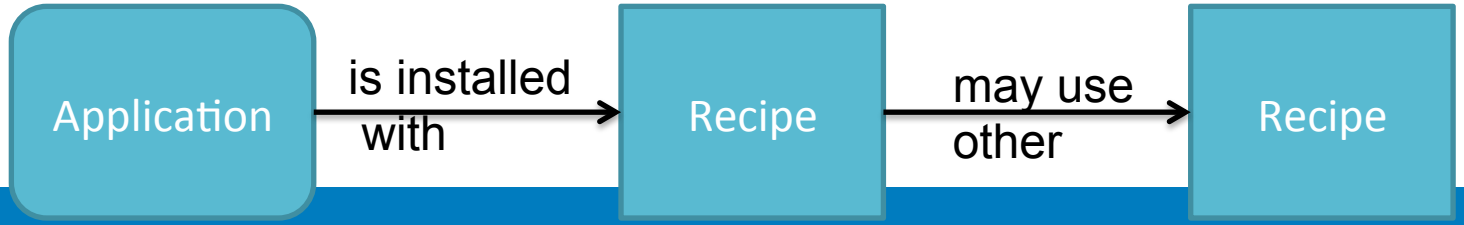


Contextualization Service

- Allows VRC managers to define:
  - applications:
    - name, version, configuration options, tags, …
  - recipes:
    - script that installs a given application
- Allows end-users to:
  - query available applications
  - request the *userdata* for installing a set of apps

Applications & Recipes

- Recipes can be:
  - a script included in the recipe definition
  - an url pointing to the script
  - a git repo where the script is contained
- Recipe receive as input when executed:
  - Application info (name, version, config)
  - Other recipes it depends on



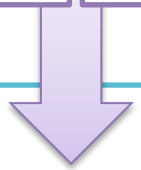
Implementation details

- Written in python with Flask
- VOMS enabled:
  - WSGI filter (similar to OpenStack VOMS module)
  - write operations require valid VO membership
- REST API for all operations
- Mongodb for storing data

Simple example

```
cat SAMPLE_APP_DATA
{
    "name": "TestApp",
    "version": “6.4.4",
    "recipe": {
        "type": "script",
        "src": "'#!/bin/sh\necho "HELLO"\n'"
    }
}
```

```
curl -k -v -X POST --cert $X509_USER_PROXY \
     -H "Accept: application/json" \
     -H "Content-type: application/json" \
     -d @SAMPLE_APP_DATA $SERVER/app
```



```
{
"apps": [
        {
            "creation": "Fri Sep 13 10:13:44 2013",
            "endorser": "/DC=es/DC=irisgrid/O=ifca/CN=Enol-Fernandez-delCastillo",
            "endorser_vo": "ops.vo.ibergrid.eu",
            "href": "/app/TestApp/6441",
            "id": "5232e558fae682063a5454e7",
            "name": "TestApp",
            "recipe": "/recipe/5232e558fae682063a5454e6",
            "version": "6.4.4"
        }
    ]
}
```

User Side

- Query available apps:
  - by name, version, tags, endorser, …
- Ask for contextualization info:
  - POST to /contextualization with:
    - id's of apps to install
    - extra configuration parameters
  - Returns cloud-init compatible user-data:
    - python script that executes the recipes in order

Simple example

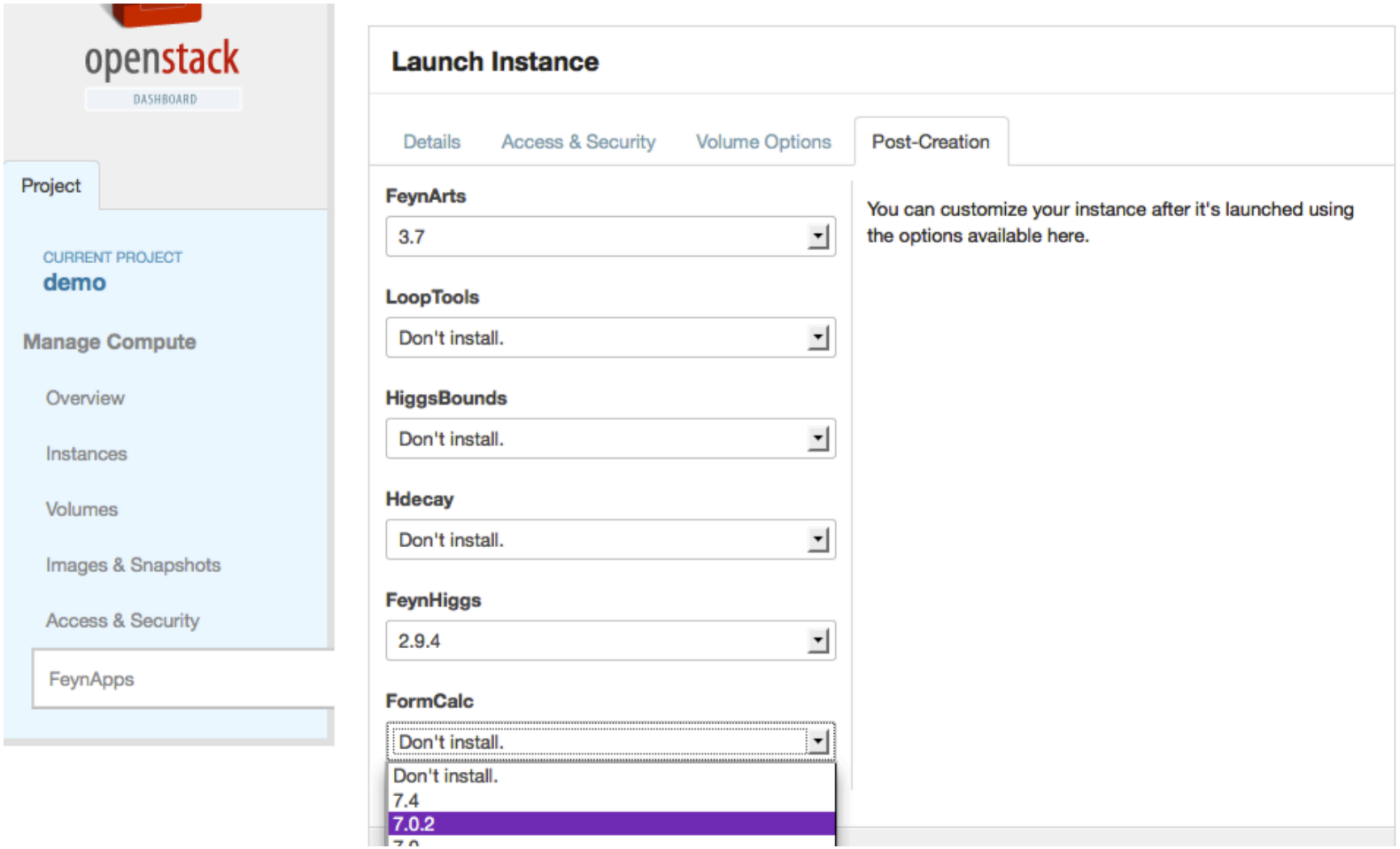
```
cat CONTEXT_REQUEST
{
   "apps": [
        {
          "id": "51ef8de311ec42540a3349c7",
          "config": {"server": ”example.com"}
        }
        …
    ],
}
```

```
curl -k -v -X POST --cert $X509_USER_PROXY \
     -H "Accept: application/json" \
     -H "Content-type: application/json" \
     -d @CONTEXT_REQUEST $SERVER/context
```

```
{
    "type": "cloud-init",
    "userdata": "<base64 enconded script>"
}
```

```
occi --endpoint $ENDPOINT --auth x509 --user-cred $X509_USER_PROXY --voms \
     --action create --resource compute \
     --mixin $TEMPLATE --mixin $RESOURCE \
     --attributes title="$NAME" \
     --context user_data=“<the script>"
```

Rewritting FeynApps



FeynApps Recipe

```
cat RECIPE_DATA
{
 "desc": "FeynAppsHelper“,
 "type": "script",
 "url": "https://github.com/enolfc/feynapps.git",
 "file": "contextualize",
 "repo-type": "git"
}
```

```
curl -k $DEBUG -X POST --cert $X509_USER_PROXY \
     -H "Accept: application/json" \
     -H "Content-type: application/json" \
     -d @RECIPE_DATA $SERVER/recipe
```

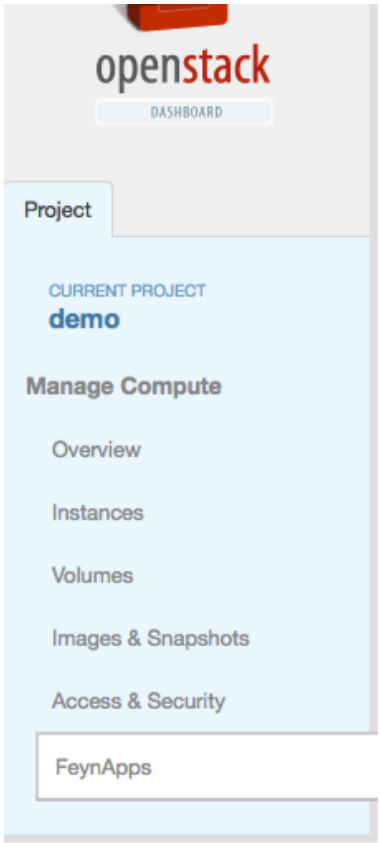

```
{
    "recipe": {
        "chain": "/recipe_chain/5232e8dffae6820639d51eac",
        "creation": "Fri Sep 13 10:28:47 2013",
        "desc": "FeynAppsHelper. Installs the Feynapps applications",
        "endorser": "/DC=es/DC=irisgrid/O=ifca/CN=Enol-Fernandez-delCastillo",
        "endorser_vo": "ops.vo.ibergrid.eu",
        "file": "contextualize",
        "href": "/recipe/5232e8dffae6820639d51eac",
        "id": "5232e8dffae6820639d51eac",
        "repo-type": "git",
        "type": "script",
        "url": "https://github.com/enolfc/feynapps.git"
    }
}
```

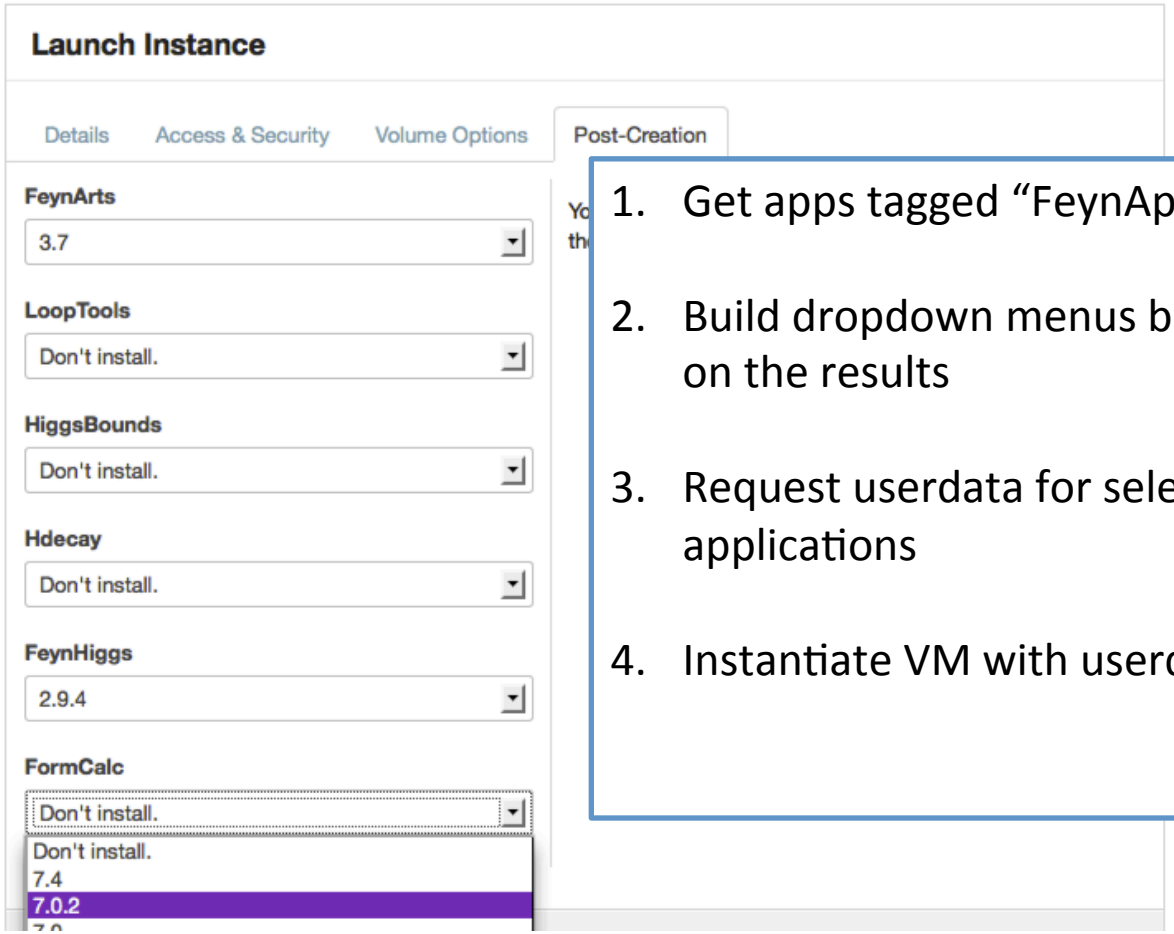
FeynApps App

```
cat APP_DATA
{
    "name": ”FormCalc",
    "version": ”7.0.3",
    "recipe": {
        "file": "installers/formcalc.sh",
        "type": "script",
        "url": "https://github.com/enolfc/feynapps.git",
        "repo-type": "git",
        "uses": ”5232e8dffae6820639d51eac"
    },
    "tags": ["FeynApps"],
    "config": {
        "base_url": "http://www.feynarts.de/formcalc/”,
        "tar_file": “FormCalc-7.0.3”
    }
}
```

```
{
    "apps": ["id": "5232e9c1fae682063a5454ed",
             "name": "Prospino",
             "recipe": "/recipe/5232e9c1fae682063a5454ec”,
             …
        }
    ]
}
```

FeynApps Panel



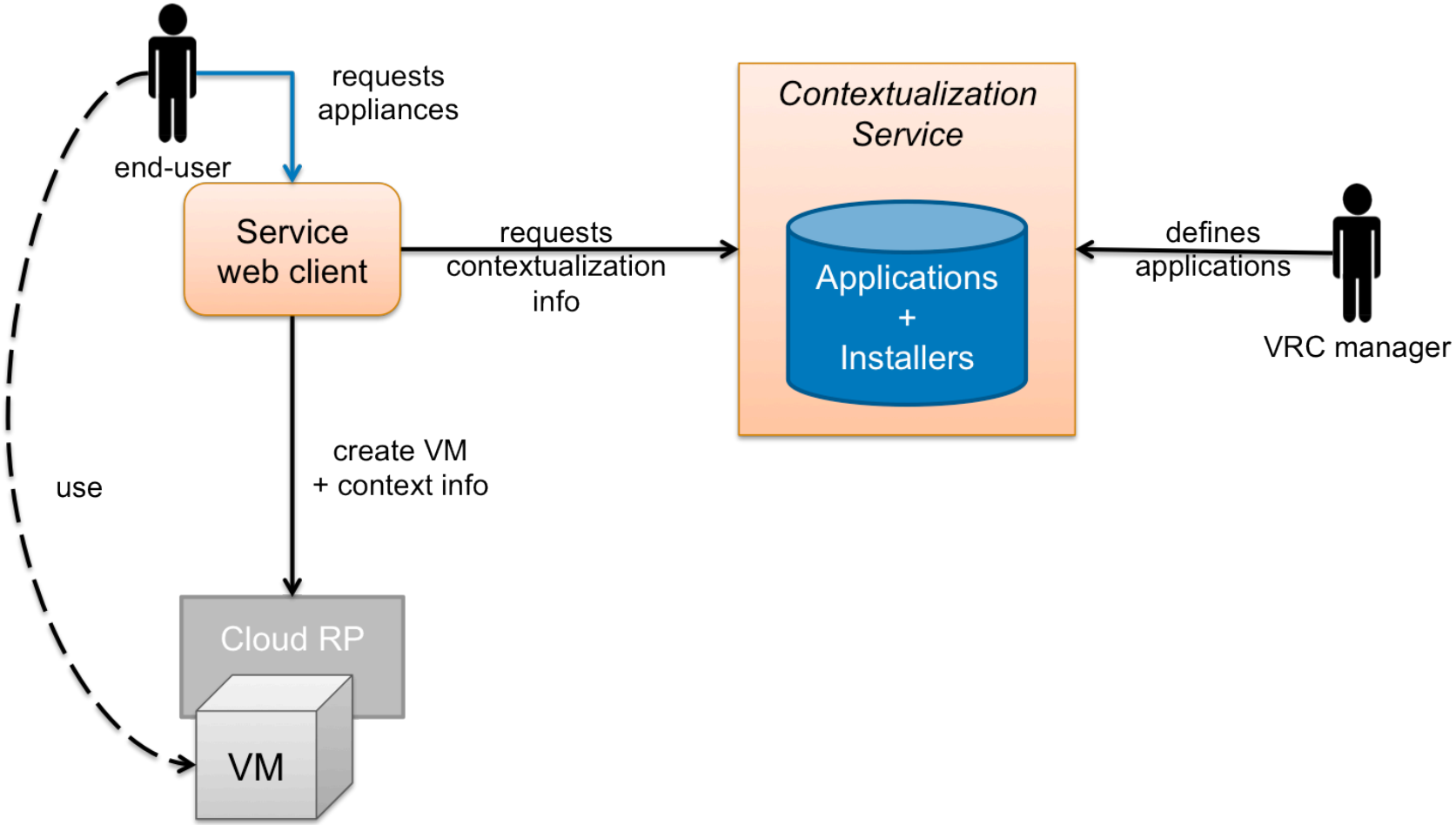


1. Get apps tagged “FeynApps”
2. Build dropdown menus based on the results
3. Request userdata for selected applications
4. Instantiate VM with userdata

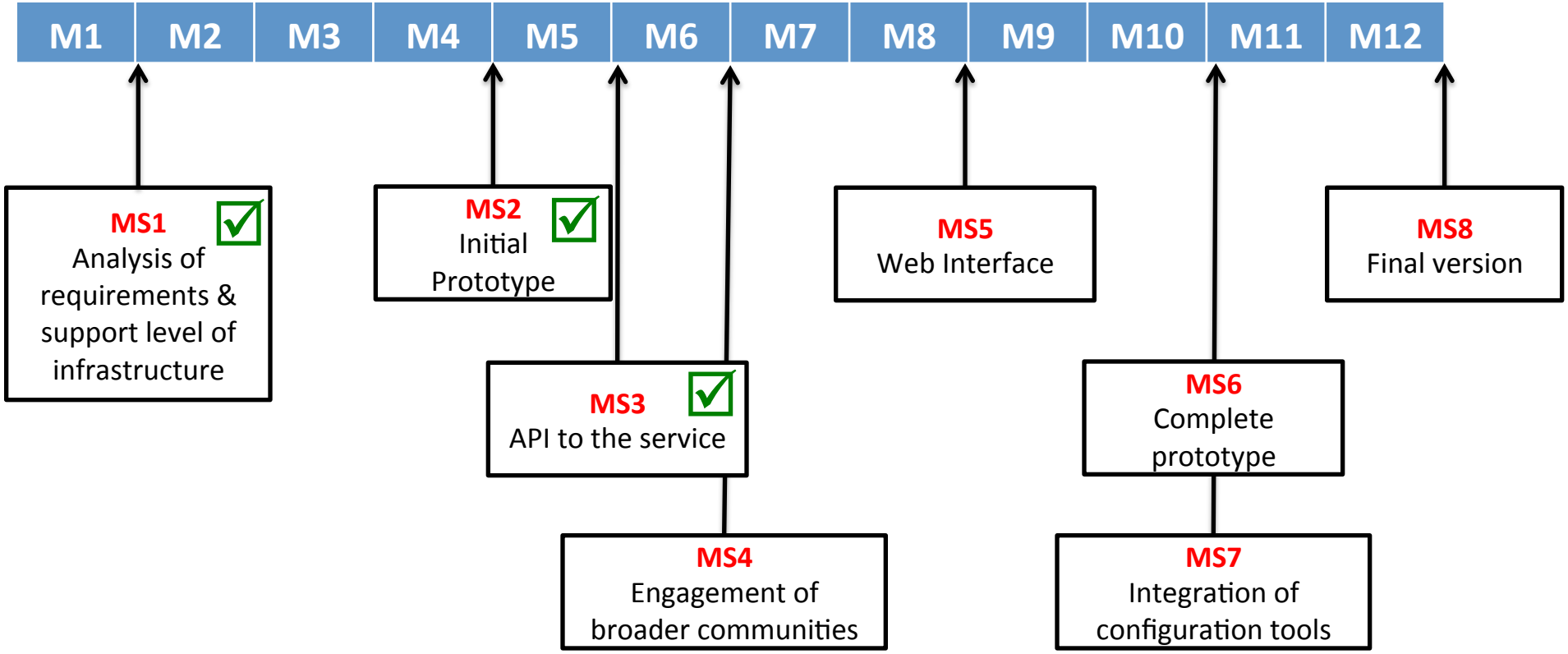
Next Steps

- API is ok to build services, but too complex for users/VRC managers:
  - working on a web GUI
- Basic recipes repository:
  - download; untar; configure; make; make install
  - puppet modules
- Get some external users

Future Iterations



Timeline


M1
M2
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
MS1
Analysis of requirements & support level of infrastructure
MS2
Initial Prototype
MS3
API to the service
MS4
Engagement of broader communities
MS5
Web Interface
MS6
Complete prototype
MS7
Integration of configuration tools
MS8
Final version

Thanks!

Automatic Deployment and Execution of Applications using Cloud Services