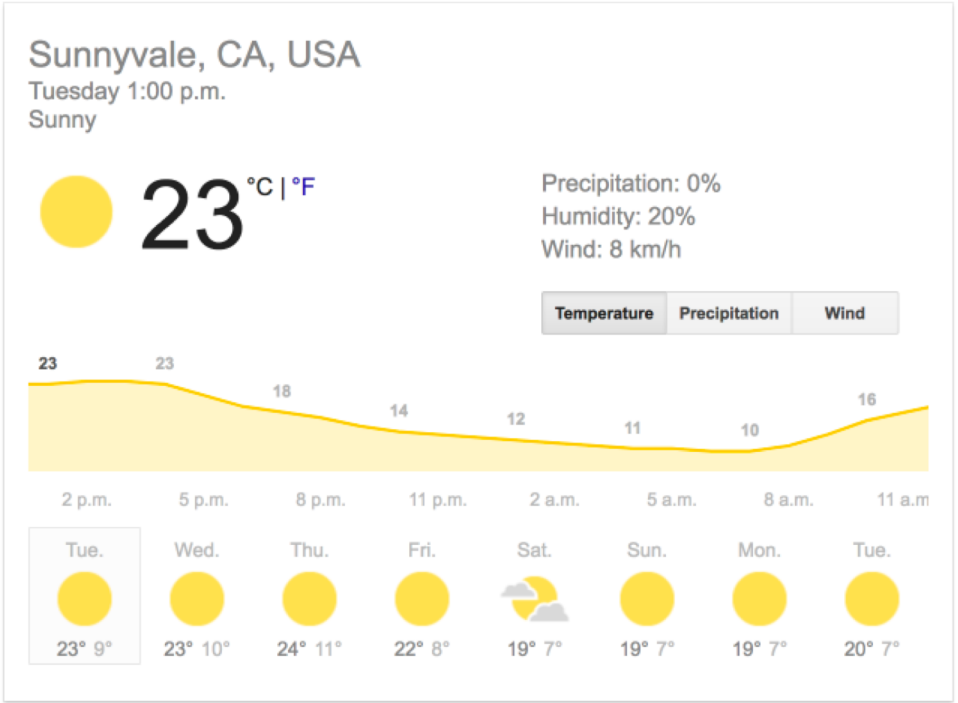
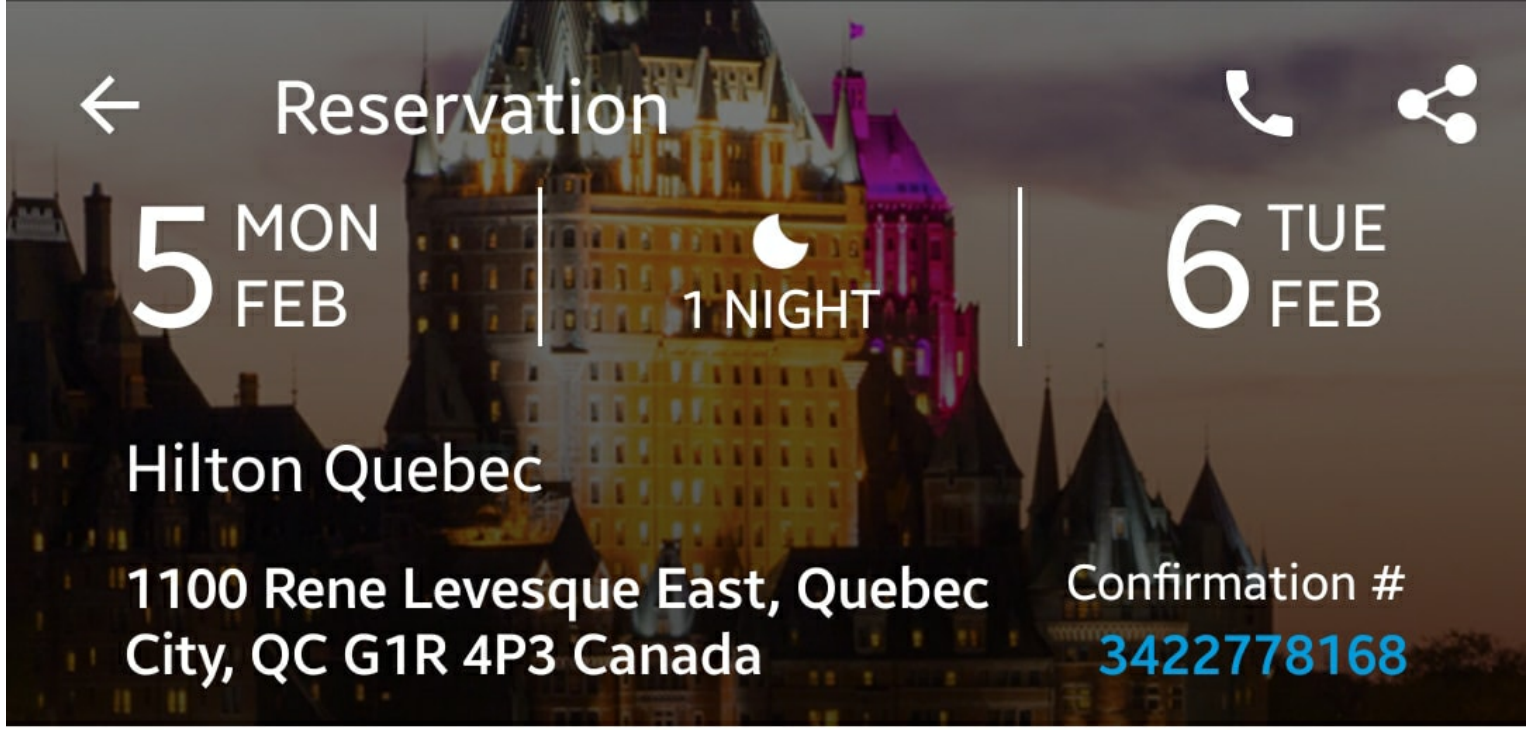
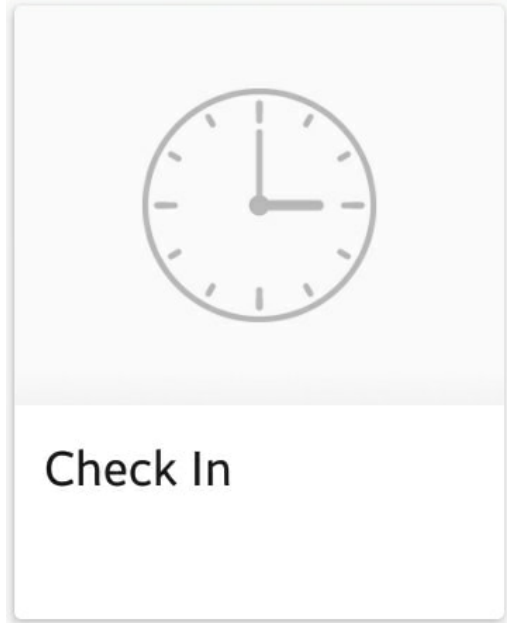
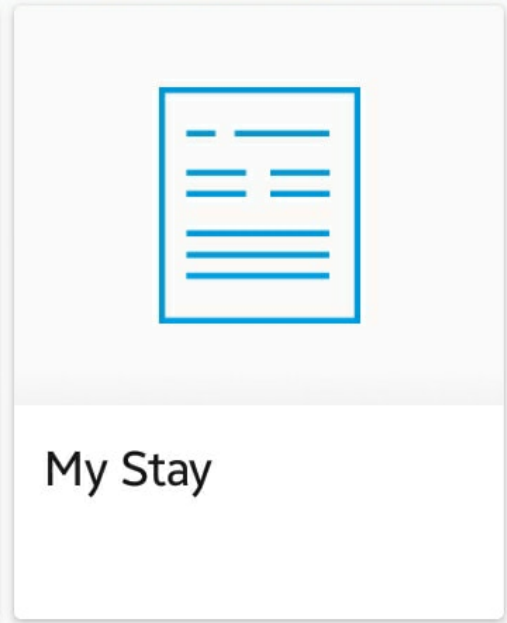
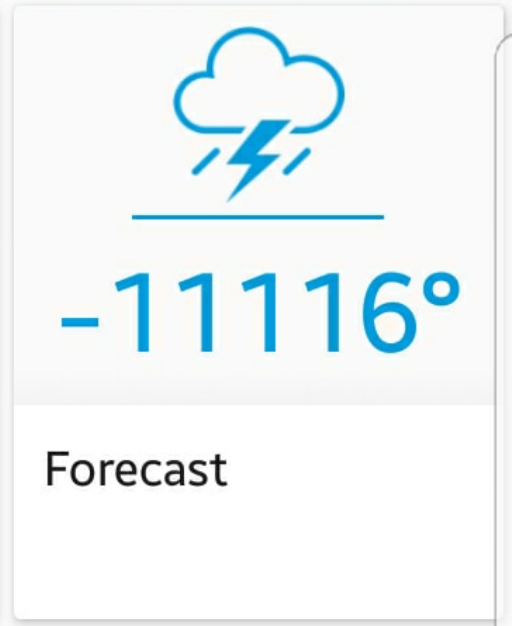
WHERE THE HELM ARE YOUR BINARIES?











JOIN.JFROG.COM

ABOUT ME

@jbaruch

CLOUD NATIVE COMPUTING FOUNDATION
AMBASSADOR

SHOWNOTES

- www.jfrog.com/shownotes
- Slides
- Video (tomorrow!)
- All the links!
- Ratings, comments
- Raffle!

WHAT THE HELM IS HELM?

Dependency manager for Kubernetes

HOW TO DEPLOY ANYTHING TO K8S

- Copy YAML
- Paste YAML
- Fix indents
- Repeat

KUBERNETES RESOURCE

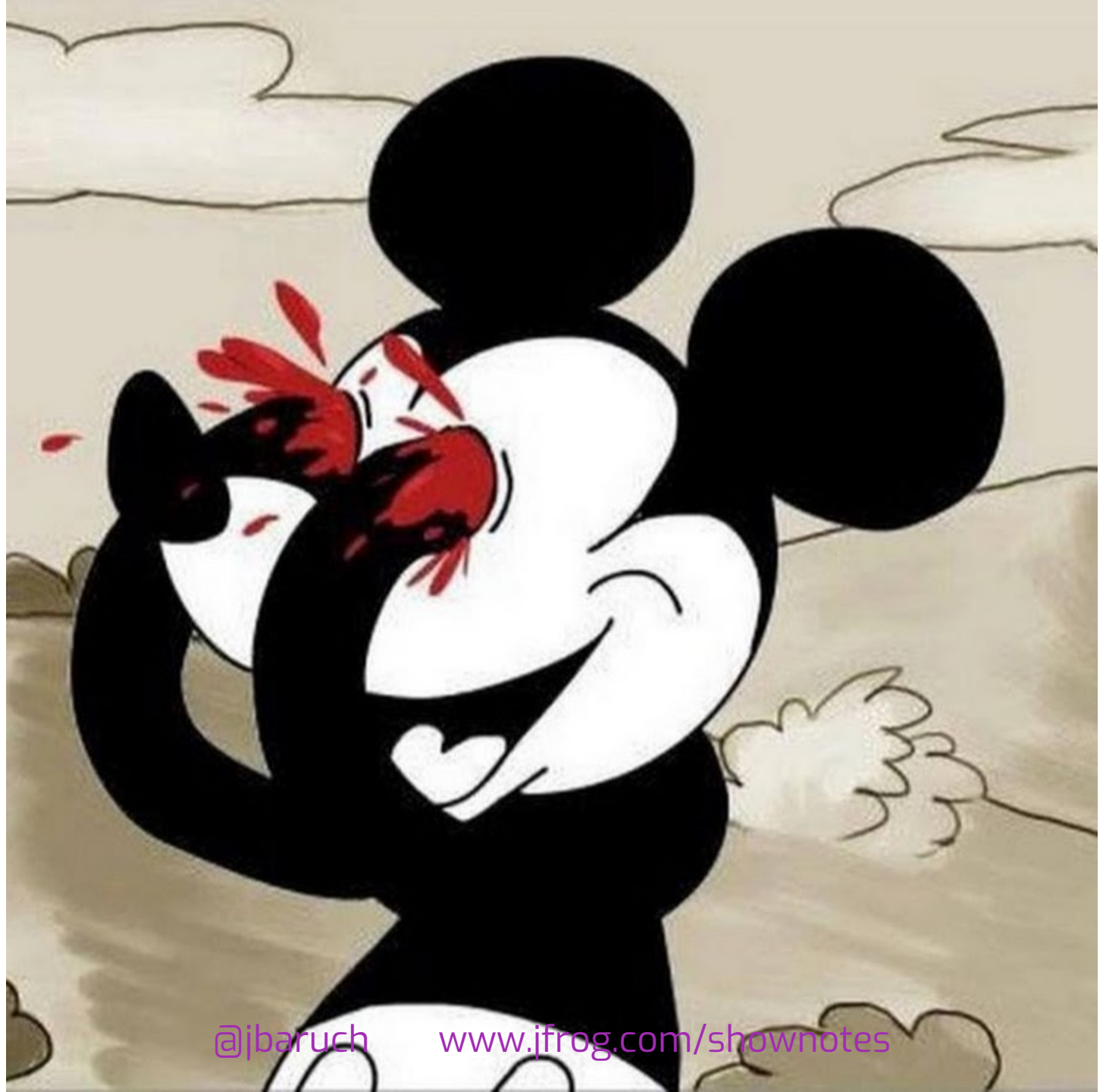
```
{
  "kind": "Deployment",
  "apiVersion": "extensions/v1beta1",
  "metadata": {
    "name": "my-release-docker-app-chart"
    },
      "spec": {
        "containers": [
          {
            "name": "docker-app-chart",
            "image": "docker.artifactory/docker-app:1.0",
```

LET'S BUILD A NEW ONE!

```
> docker build -t docker.artifactory/docker-app:1.1
```

ONE LAST THING...

```
> sed -i.bak
`s#docker.artifactory/docker-app:1.1#${imageTag}#`
deployment.yaml
```


@jbaruch www.jfrog.com/shownotes

OR JUST USE :LATEST

```
"image": "docker.artifactory/docker-app:latest"
```

ENTER HELM

ENCAPSULATED PACKAGES OF KUBERNETES DEPLOYMENTS

All this...

```
xrayxray-analysis
xray-event
xray-indexer
xray-nfs-server
xray-persist
```

Becomes this

```
xray
```

POWERFUL TEMPLATING FOR DESCRIPTOR FILES

```
{
  "kind": "Deployment",
  "apiVersion": "extensions/v1beta1",
  "metadata": {
    "name": "{{ template "docker-app.fullname" . }}"
    },
      "spec": {
        "containers": [
          {
            "name": "{{ template "docker-app.name" . }}",
            "image": "{{ .Values.image.repository }}:
                      {{ .Values.image.tag }}"
```

VALUES:

```yaml
# Default values for docker-app.
# This is a YAML-formatted file.
# Declare name/value pairs to be passed into your templates.

image:
  repository: docker.artifactory/docker-app
  tag: 1.1
  secretName: regsecret
  pullPolicy: Always
```

SIMPLE!

- Templates
- Values
- Metadata

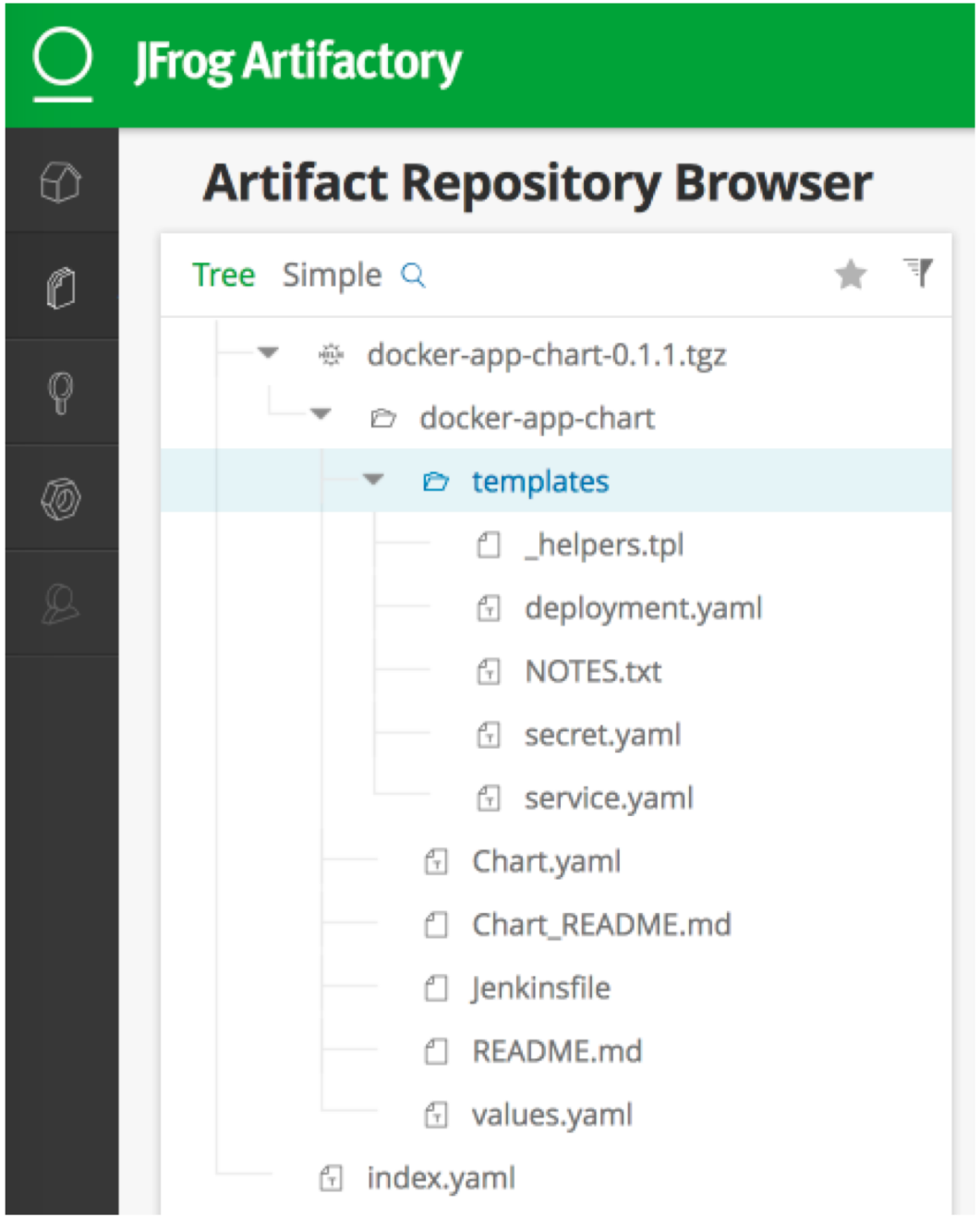
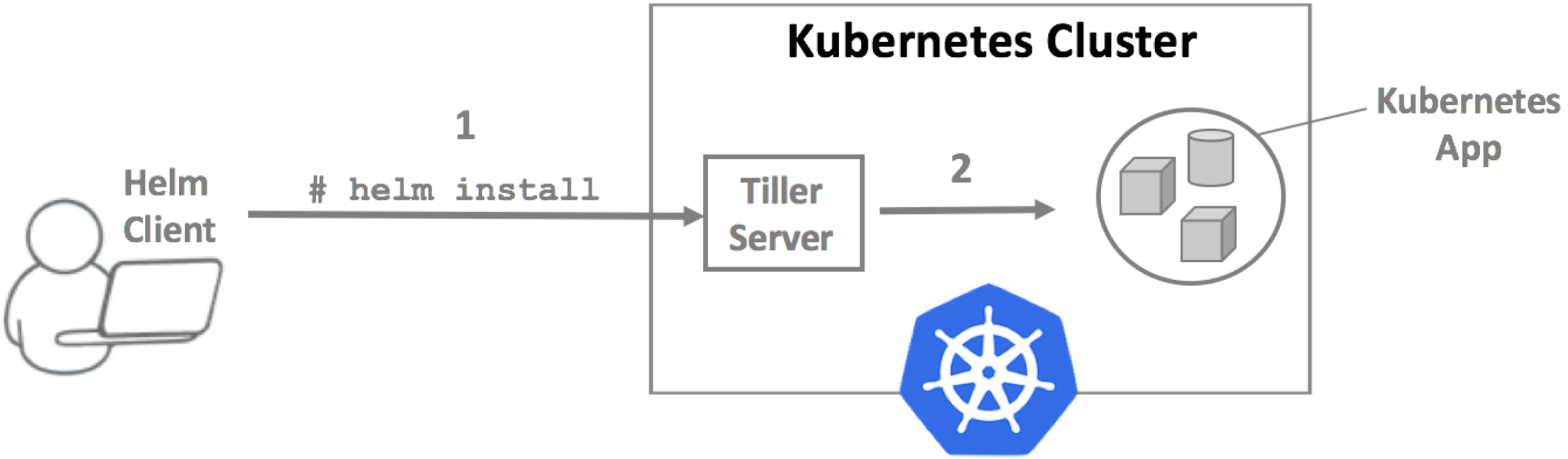


CHART <-> IMAGE RELATIONSHIP

- Using templates we can reuse charts for multiple image versions
- Chart versions != Image versions

KUBERNETES CLUSTER CONTROL



TWO PARTS

Helm client

- Local chart development
- Managing repositories
- Interacting with the Tiller server

Tiller Server

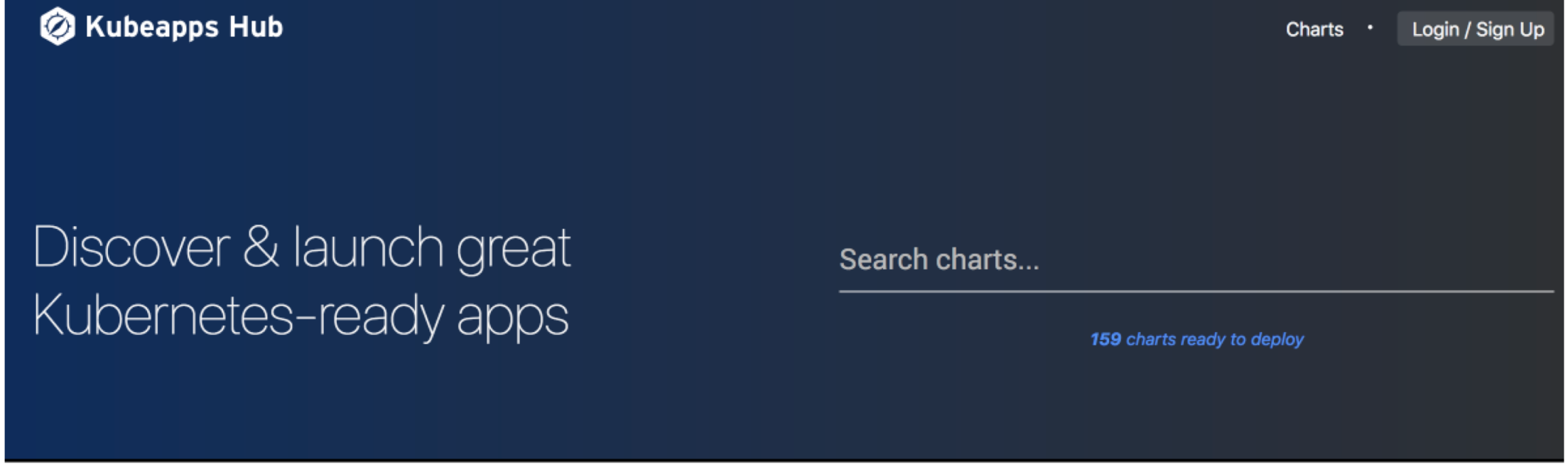
- Listening for incoming requests from the Helm client
- Combining a chart and configuration to build a release
- Installing charts into Kubernetes, and then tracking the subsequent release
- Upgrading and uninstalling charts by interacting with Kubernetes

HELM COMMANDS

```
>helm init
>helm search
>helm install
>helm status
>helm repo
```

HELM REPOSITORIES

- Official repository - kubeapps.com



acs-engine-autoscaler
2.1.1 stable ☆ 0

aerospike
v3.14.1.2 stable ☆ 0

anchore-engine
0.1.6 stable ☆ 0

artifactory
5.8.3 stable ☆ 17

buildkite
3 stable ☆ 1

burrow
0.17.1 incubator ☆ 0

cassandra
0.7.3 incubator ☆ 3

centrifugo
1.7.3 stable ☆ 0

HELM REPOSITORIES

- Official repository - kubeapps.com
- Get a local one!
- Option 1: Create your own:
  - Run an http server with index.yaml
  - Run `helm repo index` to generate one the index
- Option 2: Use JFrog Artifactory
  - Universal Artifact Repository which supports Docker, Helm and everything else

WHAT DEPENDENCY MANAGERS AND PRINTERS HAVE IN COMMON?

Why I Believe

Printers

Were Sent From

Hell

To Make Us Miserable

By The Oatmeal http://theoatmeal.com

So you want to write a package manager

sam boyer
@sdboyer
systems | people
Detroit metro
Joined December 2008

You woke up this morning, rolled out of bed, and thought, "Y'know what? I don't have enough misery and suffering in my life. I know what to do—I'll write a language package manager!"

...

Package management is awful, you should quit right now

Package management is a nasty domain. Really nasty. On the surface, it *seems* like a purely technical problem, amenable to purely technical solutions. And so, quite reasonably, people approach it that way. Over time, these folks move inexorably towards the conclusion that:

1. software is terrible
2. people are terrible
3. there are too many different scenarios
4. nothing will really work for sure
5. it's provable that nothing will really work for sure
6. our lives are meaningless perturbations in a swirling vortex of chaos and entropy

★ Top highlight

SEVEN DEADLY SINS OF PACKAGE MANAGERS

1. Over-architecture
2. Lack of planning for enterprise scenarios (e.g., no private repositories, won't scale, etc.)
3. Downloadable index
4. CSDR (cross-site dependency resolution) loopholes
5. Lack of proper package authentication
6. Lack of version management
7. Using the wrong service for the central registry (and hard-coding it)

SEVEN DEADLY SINS OF PACKAGE MANAGERS

1. Over-architecture
2. Lack of planning for enterprise scenarios (e.g., no private repositories, won't scale, etc.)
3. Downloadable index
4. ~~CSDR (cross-site dependency resolution) loopholes~~
5. ~~Lack of proper package authentication~~
6. ~~Lack of version management~~
7. ~~Using the wrong service for the central registry (and hard-coding it~~)

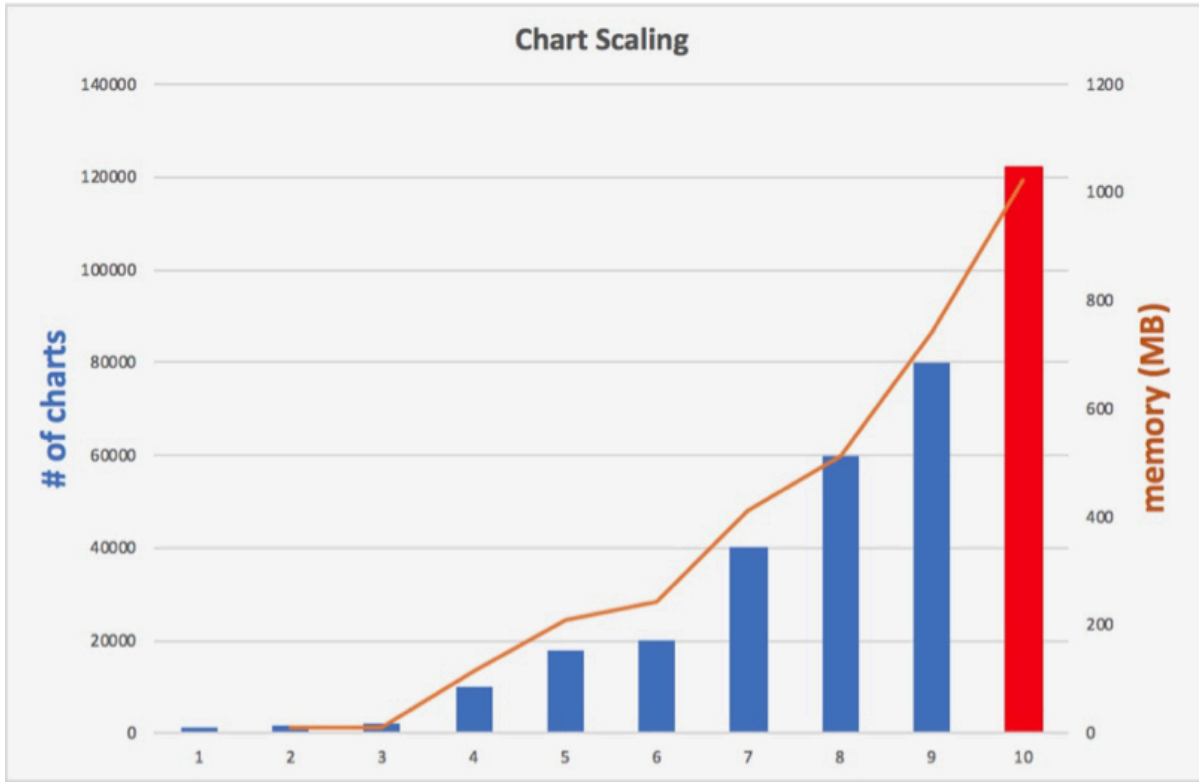
TILLER IS A PAIN

- Back in the days we had a good reason for Tiller
  - Pre-RBAC days in k8s
- Now it is just PITA
- Good news: you can use Helm without Tiller today!
- Also, Helm 3 is tillerless

DOWNLOADABLE INDEX

@jbaruch www.jfrog.com/shownotes

HOW BAD CAN IT BE?



WHY IN THE WORLD YOU NEED 120K OR CHARTS?!

VALUES:

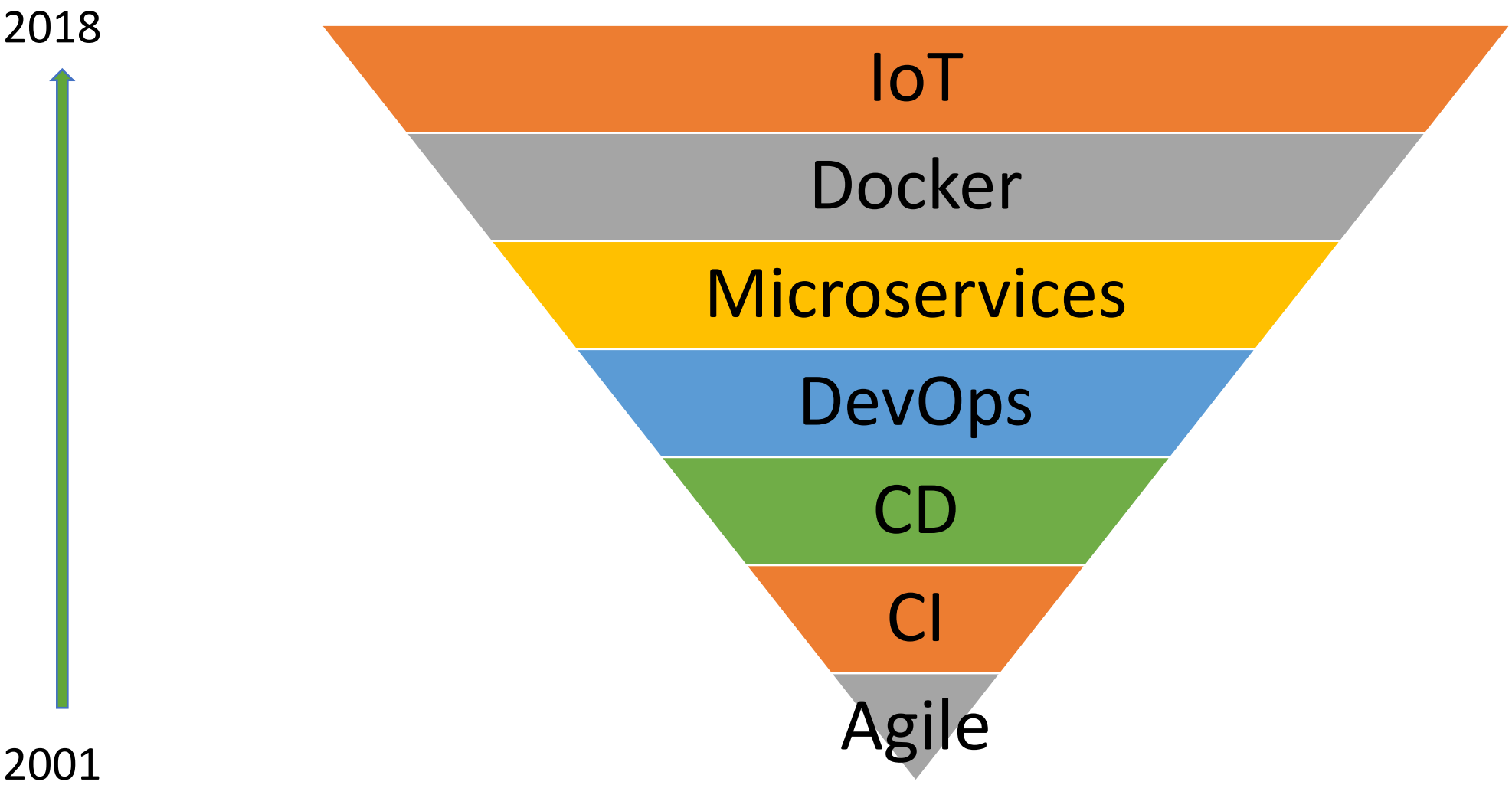
```
# Default values for docker-app.
# This is a YAML-formatted file.
# Declare name/value pairs to be passed into your templates.

image:
  repository: docker.artifactory/docker-app
  tag: 1.1
  secretName: regsecret
  pullPolicy: Always
```

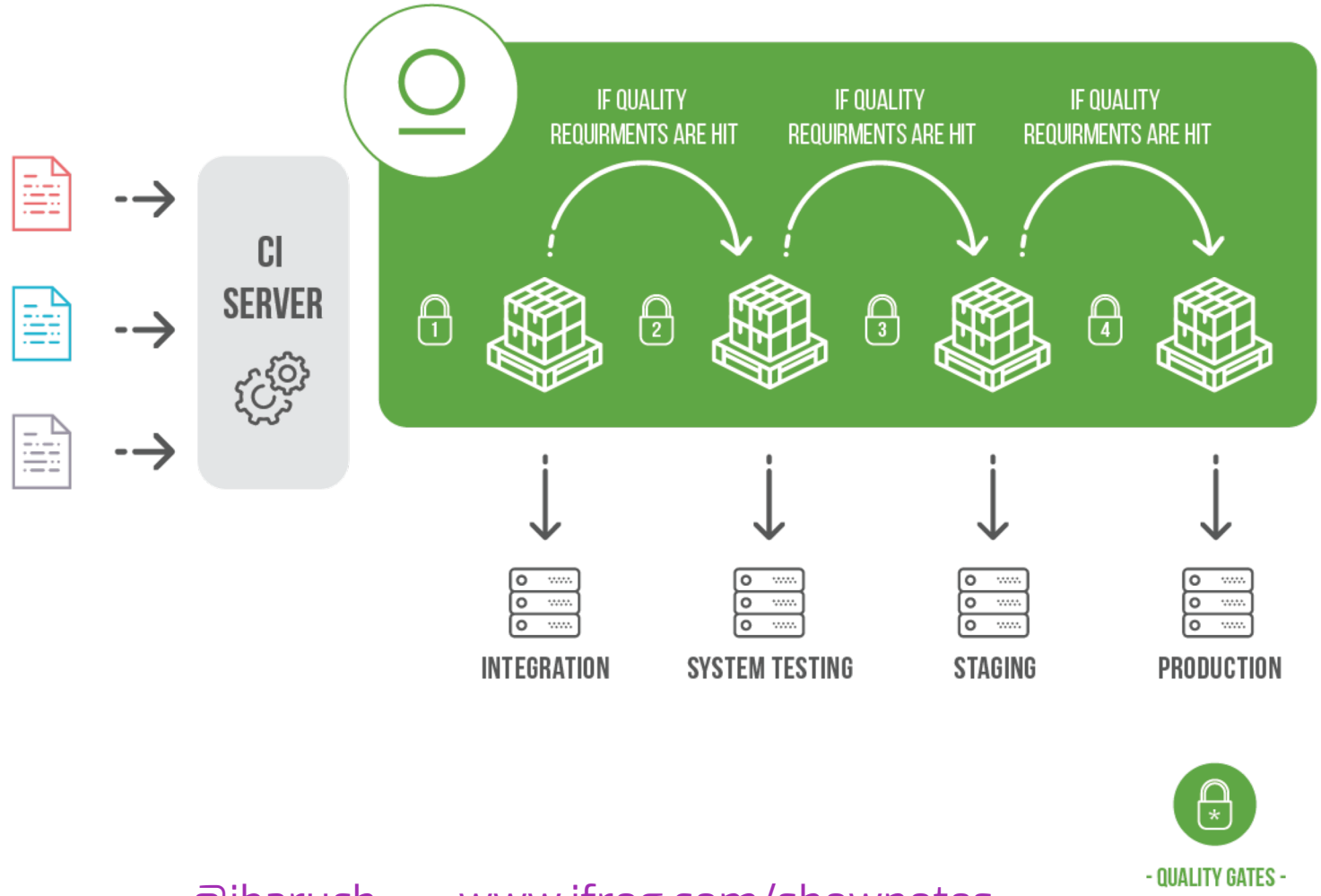
USE ARTIFACTORY:
TAKE YOUR METADATA SERIOUSLY

Where the helm is my binary?

THE AGE OF BINARIES



WHO CARES ABOUT THOSE BINARIES ANYWAY?




Dev. Team


Dev. Team
1
Declare new dependencies
Build Tools / Dependency Managers
NuGet
Ruby
GO
MSBuild
BOWER
fpm
Gradle
git
HELM
COCOAPODS
docker
eclipse
debian
VAGRANT
npm
CHEF
maven
python
ivy
puppet
sbt


DEPENDENCY MANAGERS
LIGHTYEAR
DEPENDENCY MANAGERS EVERYWHERE

```
FROM ubuntu

RUN apt-get install -y software-properties-common python
RUN apt-get install -y nodejs
RUN mvn install
RUN download_random_shit_from_internetz.sh

RUN mkdir /var/www

ADD app.js /var/www/app.js

CMD ["/usr/bin/node", "/var/www/app.js"]
```

```
FROM ubuntu
```


```
RUN apt-get install -y software-properties-common python
RUN apt-get install -y nodejs
RUN mvn install
RUN download_random_shit_from_internetz.sh

RUN mkdir /var/www

ADD app.js /var/www/app.js

CMD ["/usr/bin/node", "/var/www/app.js"]
```

```
FROM ubuntu

RUN apt-get install -y software-properties-common python
RUN apt-get install -y nodejs
RUN mvn install
RUN download_random_shit_from_internetz.sh

RUN mkdir /var/www

ADD app.js /var/www/app.js

CMD ["/usr/bin/node", "/var/www/app.js"]
```

Dependency manager

Dependency Manager

```
FROM ubuntu

RUN apt-get install -y software-properties-common python
RUN apt-get install -y nodejs
RUN mvn install
RUN download_random_shit_from_internetz.sh

RUN mkdir /var/www

ADD app.js /var/www/app.js

CMD ["/usr/bin/node", "/var/www/app.js"]
```

Dependency manager

Dependency Manager

"Make me do it!" Dependency Manager

```
FROM ubuntu

RUN apt-get install -y software-properties-common python
RUN apt-get install -y nodejs
RUN mvn install
RUN download_random_shit_from_internetz.sh

RUN mkdir /var/www

ADD app.js /var/www/app.js

CMD ["/usr/bin/node", "/var/www/app.js"]
```

Dependency manager

Dependency Manager

"Make me do it!" Dependency Manager

As good as any other Dependency Manager

```
FROM ubuntu

RUN apt-get install -y software-properties-common python
RUN apt-get install -y nodejs
RUN mvn install
RUN download_random_shit_from_internetz.sh

RUN mkdir /var/www

ADD app.js /var/www/app.js

CMD ["/usr/bin/node", "/var/www/app.js"]
```

Dependency manager

Dependency Manager

"Make me do it!" Dependency Manager

As good as any other Dependency Manager

Probably not a Dependency Manager (but who knows)

```
FROM ubuntu

RUN apt-get install -y software-properties-common python
RUN apt-get install -y nodejs
RUN mvn install
RUN download_random_shit_from_internetz.sh

RUN mkdir /var/www

ADD app.js /var/www/app.js

CMD ["/usr/bin/node", "/var/www/app.js"]
```

Dependency manager

Dependency Manager

"Make me do it!" Dependency Manager

As good as any other Dependency Manager

Probably not a Dependency Manager (but who knows)

It does bring a file from the outside, so...

```
FROM ubuntu

RUN apt-get install -y software-properties-common python
RUN apt-get install -y nodejs
RUN mvn install
RUN download_random_shit_from_internetz.sh

RUN mkdir /var/www

ADD app.js /var/www/app.js

CMD ["/usr/bin/node", "/var/www/app.js"]
```

Dependency manager

Dependency Manager

"Make me do it!" Dependency Manager

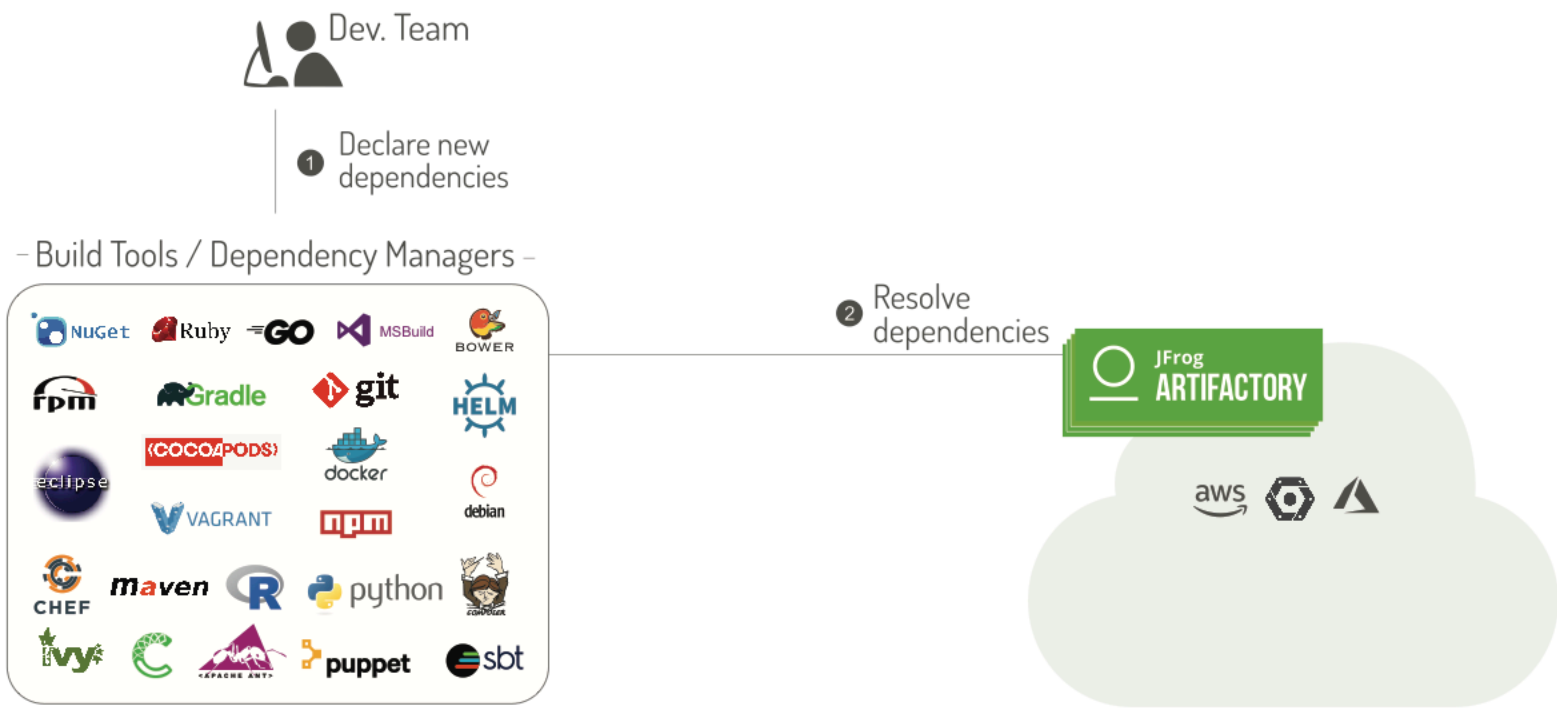
As good as any other Dependency Manager

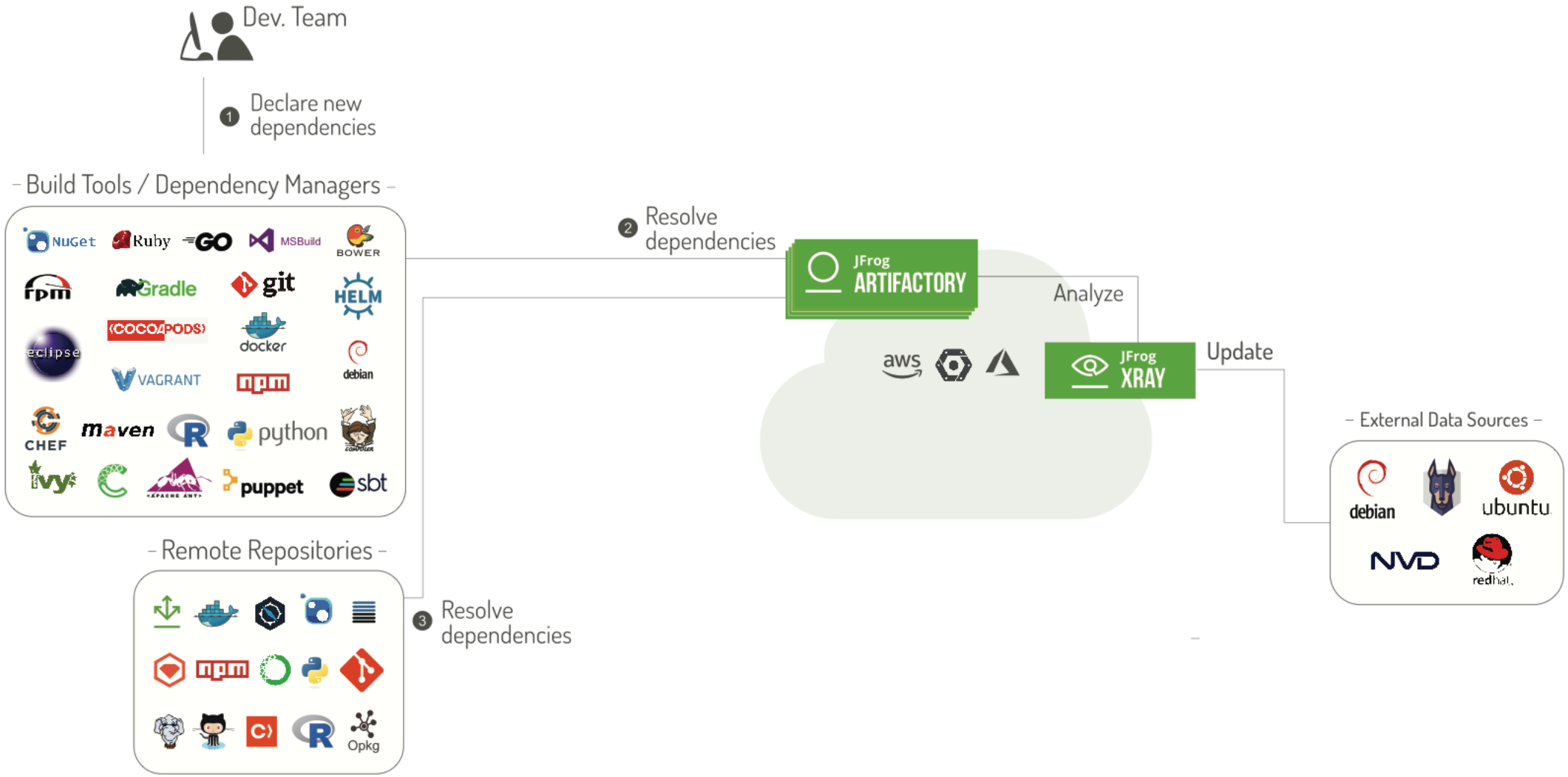
Probably not a Dependency Manager (but who knows)

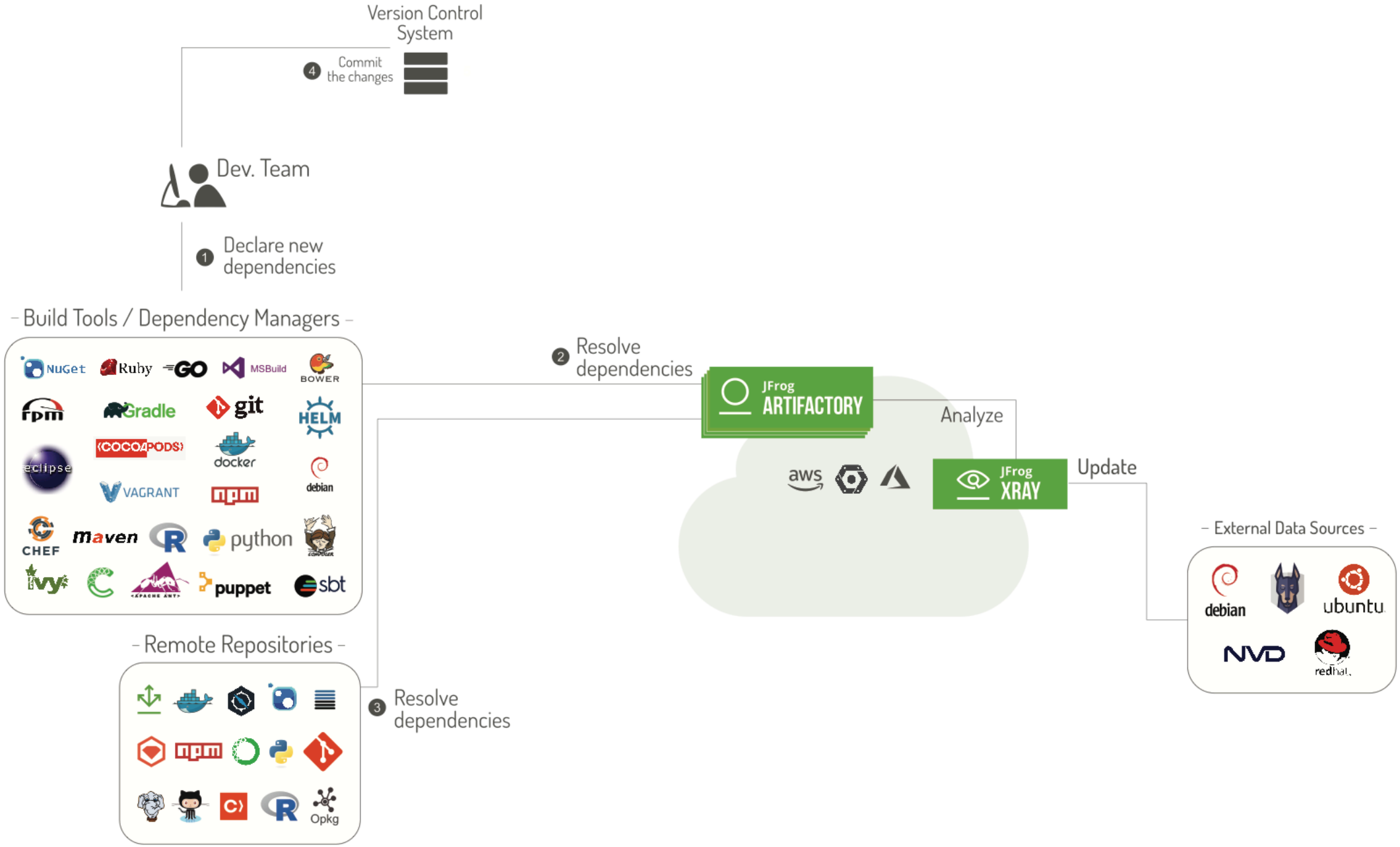
It does bring a file from the outside, so...

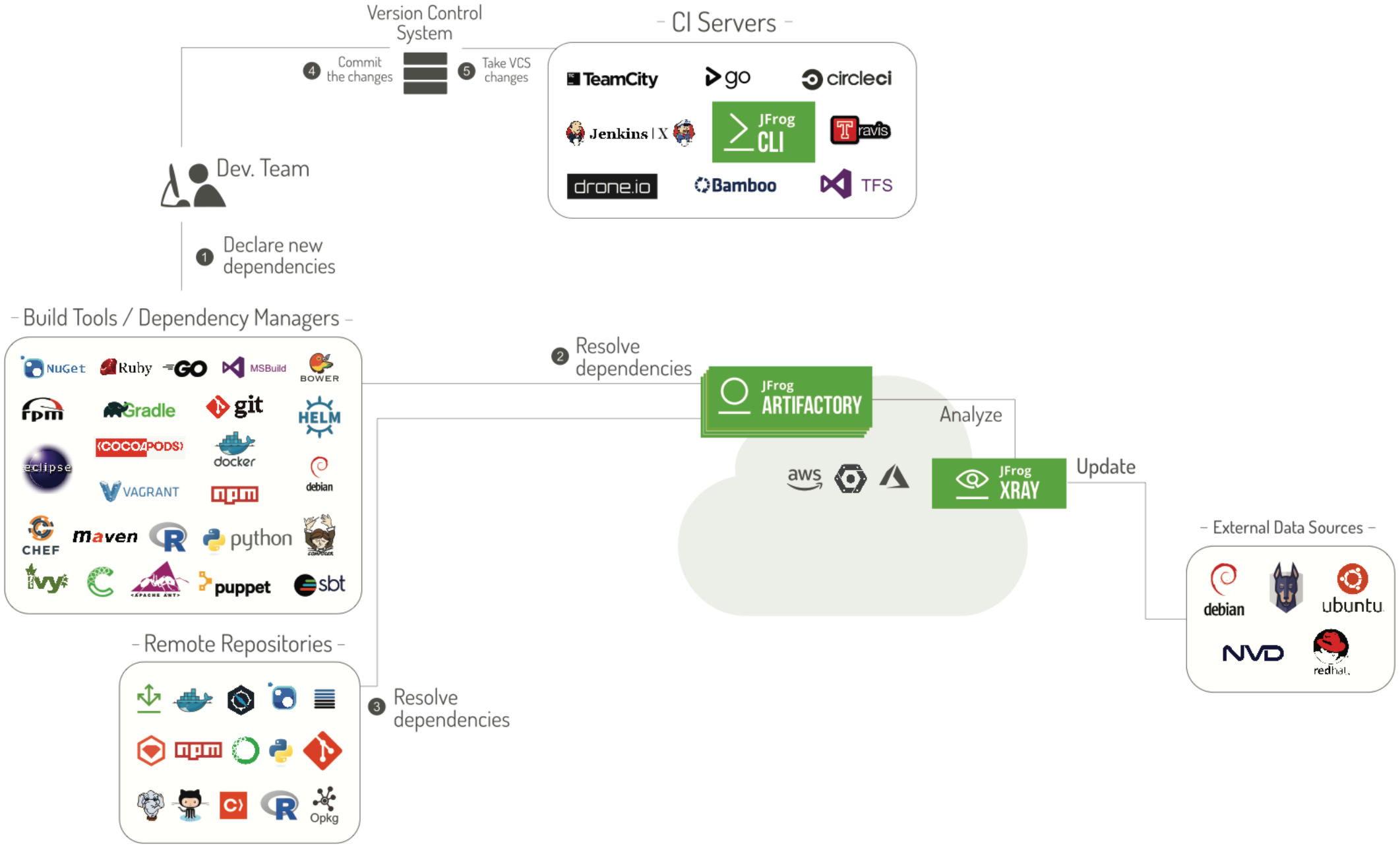
God knows what app.js does, but I bet it is a Dependency Manager of some kind

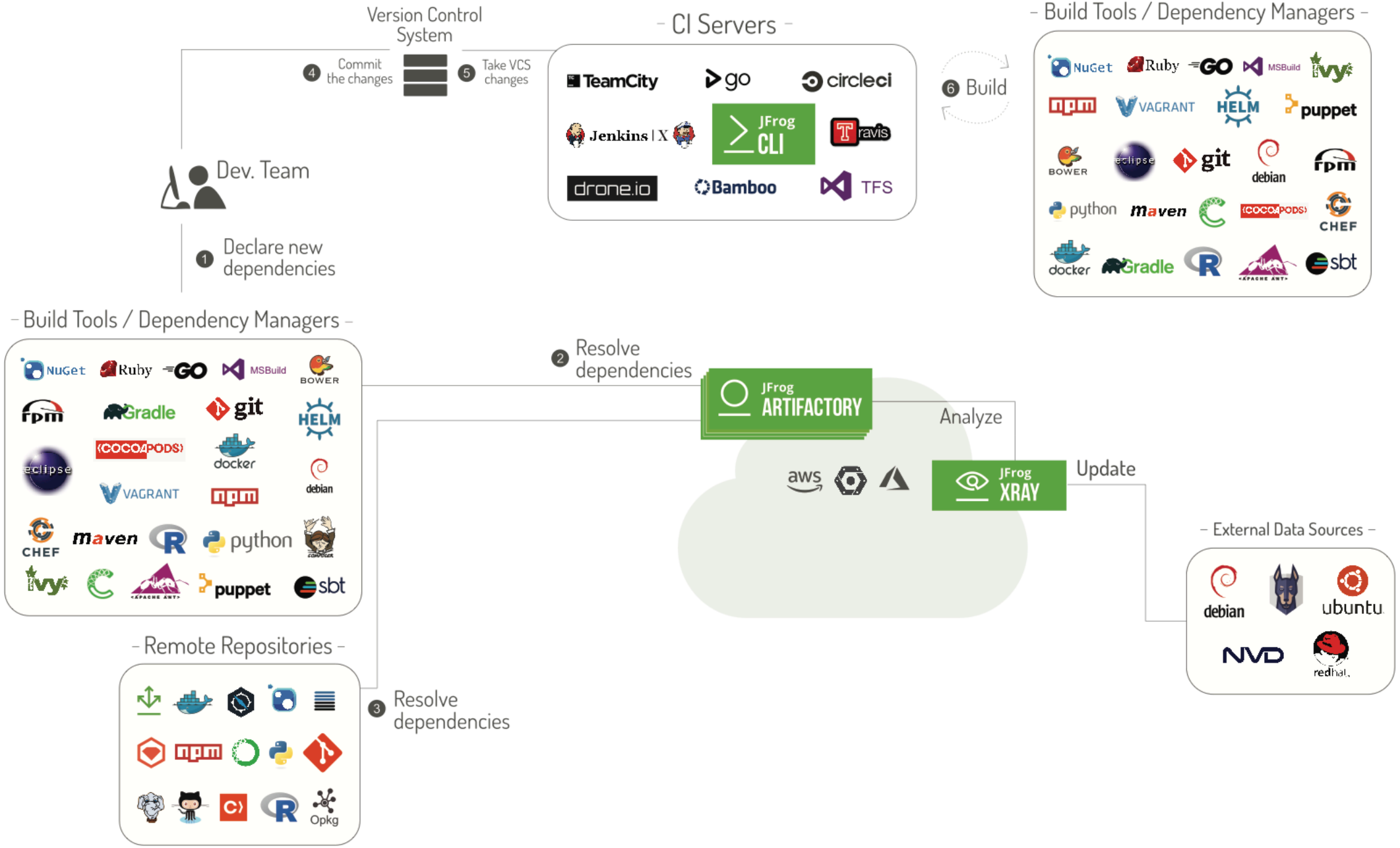

Dev. Team
1
Declare new dependencies
Build Tools / Dependency Managers
NuGet
Ruby
GO
MSBuild
BOWER
fpm
Gradle
git
HELM
COCOAPODS
docker
eclipse
debian
VAGRANT
npm
CHEF
maven
python
ivy
puppet
sbt

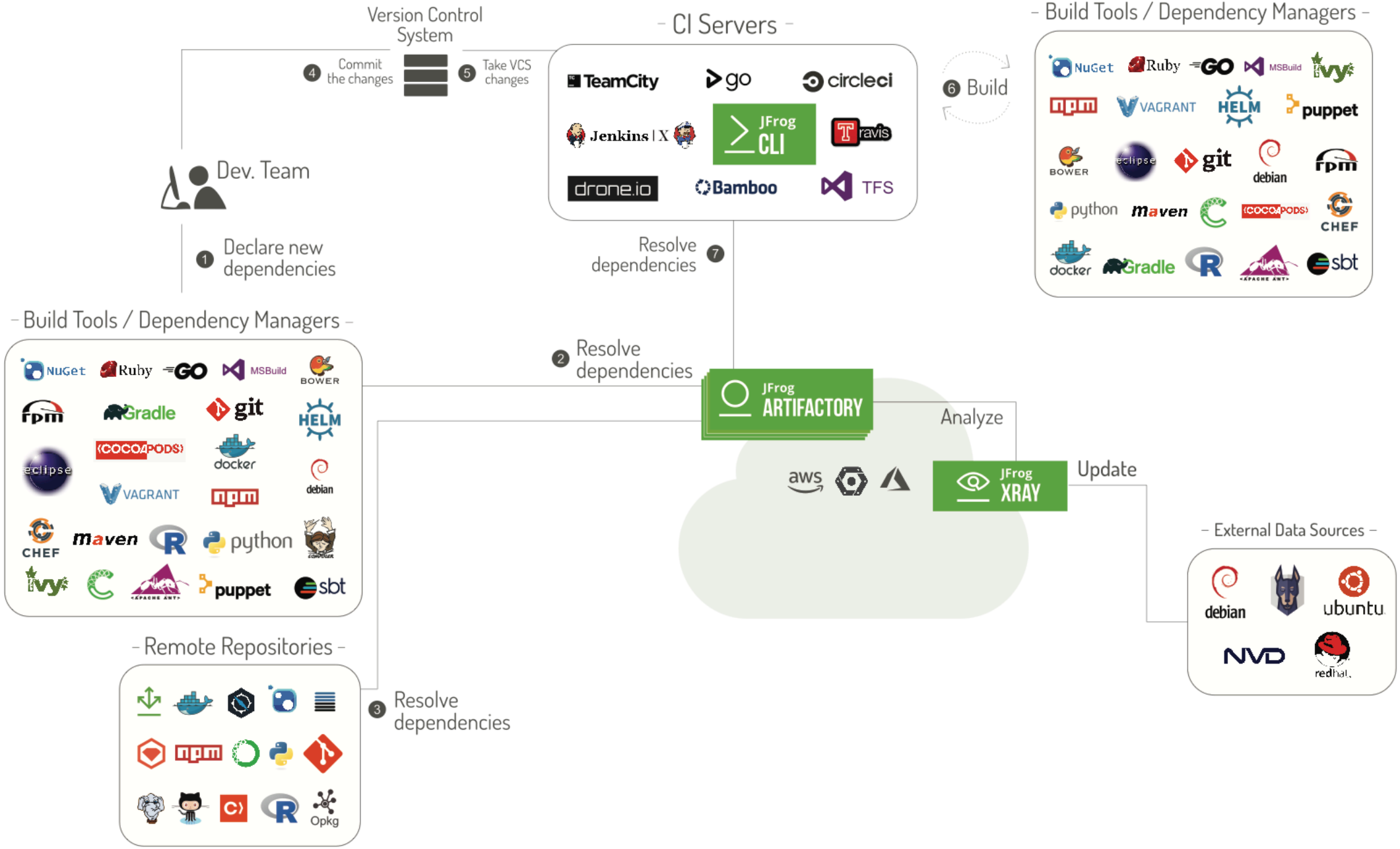

Dev. Team
1 Declare new dependencies
Build Tools / Dependency Managers
NuGet
Ruby
GO
MSBuild
BOWER
rpm
Gradle
git
HELM
COCOAPODS
eclipse
docker
debian
VAGRANT
npm
CHEF
maven
R
python
ivy
puppet
sbt
2 Resolve dependencies
JFrog
ARTIFACTORY
aws

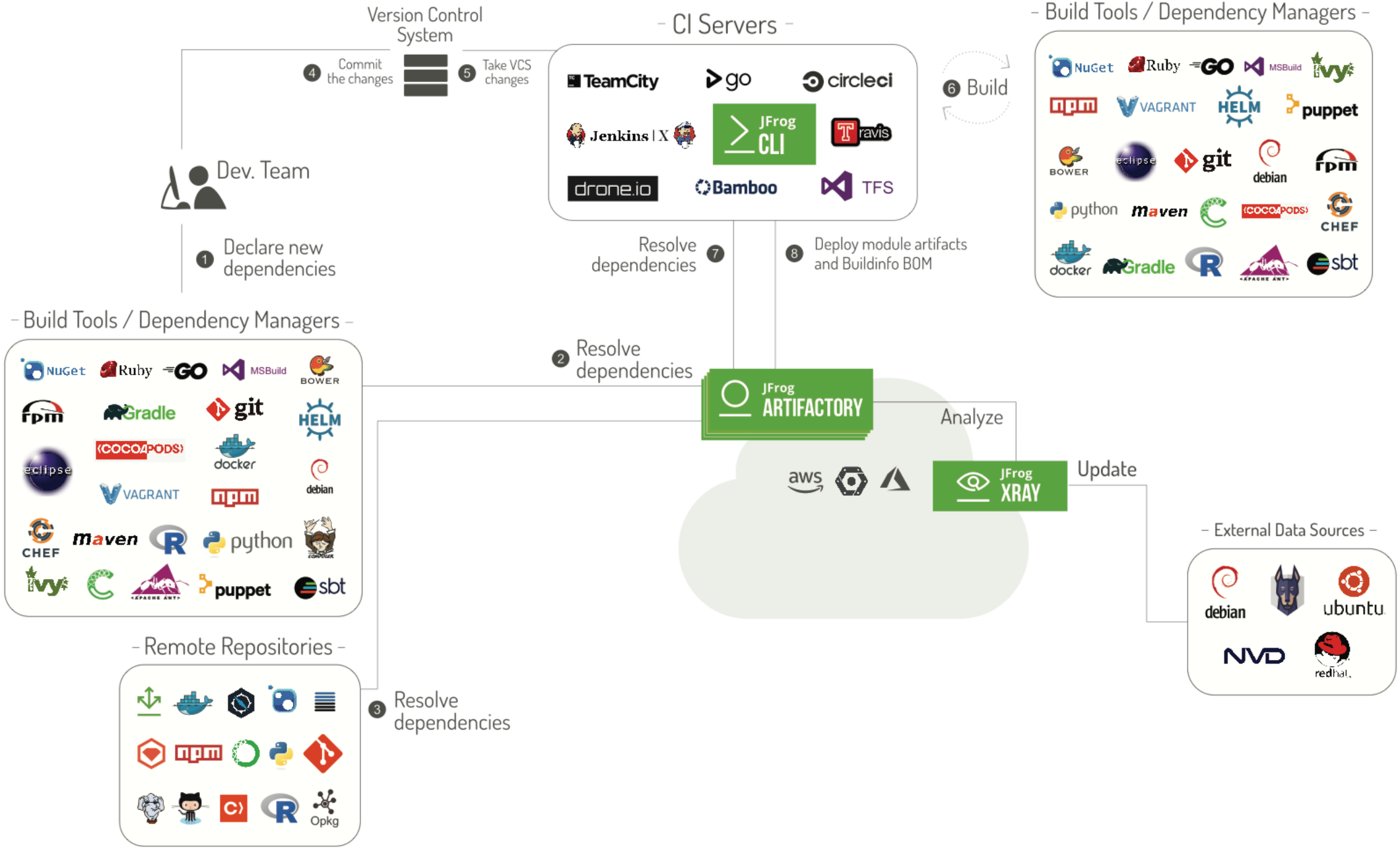

Dev. Team
1 Declare new dependencies
– Build Tools / Dependency Managers –
NuGet
Ruby
GO
MSBuild
BOWER
fpm
Gradle
git
HELM
eclipse
COCOAPODS
docker
debian
VAGRANT
npm
CHEF
maven
python
ivy
puppet
sbt
2 Resolve dependencies
JFrog ARTIFACTORY
Analyze
aws
JFrog XRAY
Update
– External Data Sources –
debian
ubuntu
NVD
redhat.
– Remote Repositories –
npm
Opkg
3 Resolve dependencies

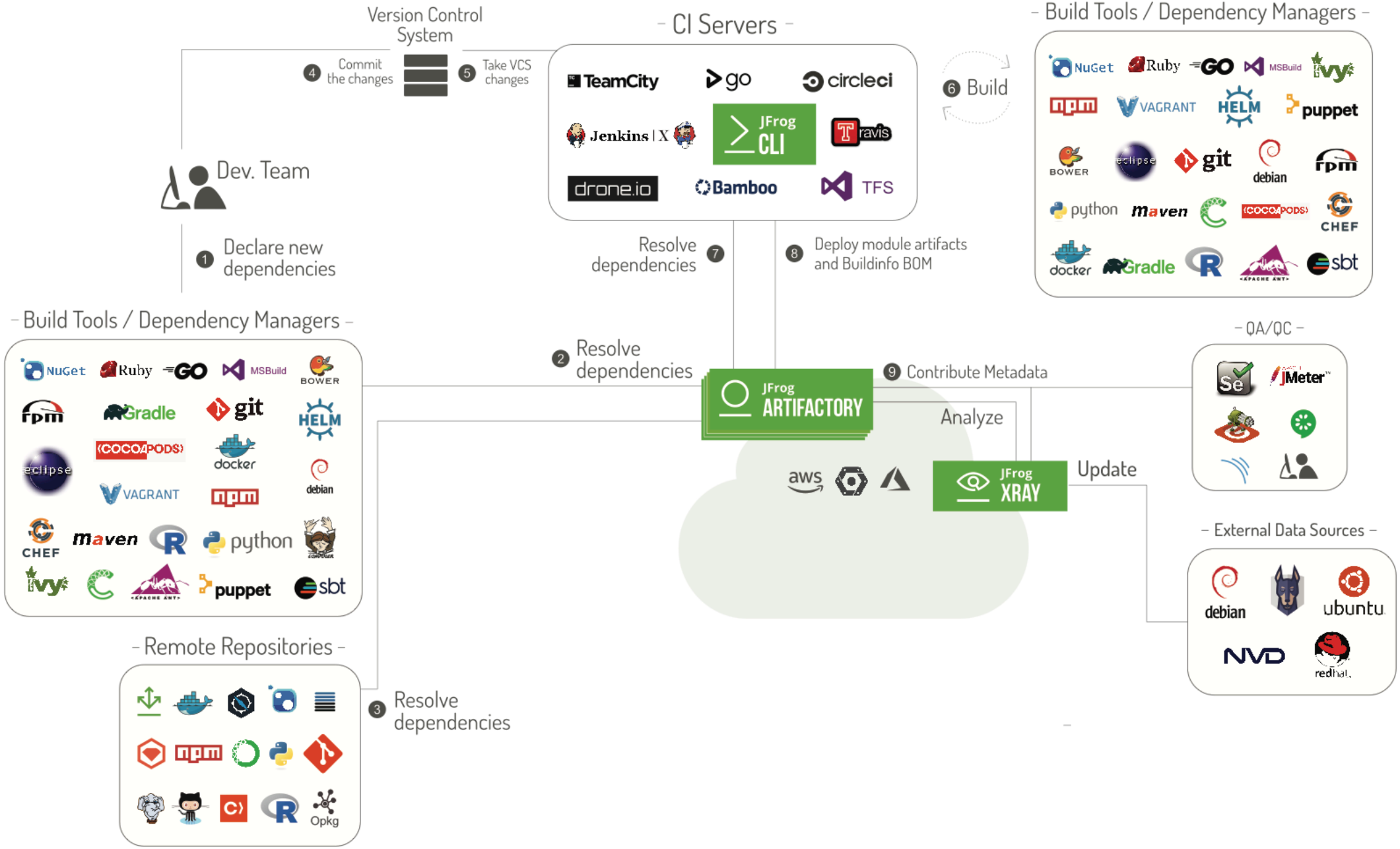

Version Control System
4 Commit the changes
Dev. Team
1 Declare new dependencies
– Build Tools / Dependency Managers –
NuGet
Ruby
GO
MSBuild
BOWER
fpm
Gradle
git
HELM
eclipse
COCOAPODS
docker
debian
VAGRANT
npm
CHEF
maven
python
ivy
puppet
sbt
2 Resolve dependencies
JFrog ARTIFACTORY
Analyze
aws
JFrog XRAY
Update
– External Data Sources –
debian
ubuntu
NVD
redhat.
– Remote Repositories –
npm
Opkg
3 Resolve dependencies

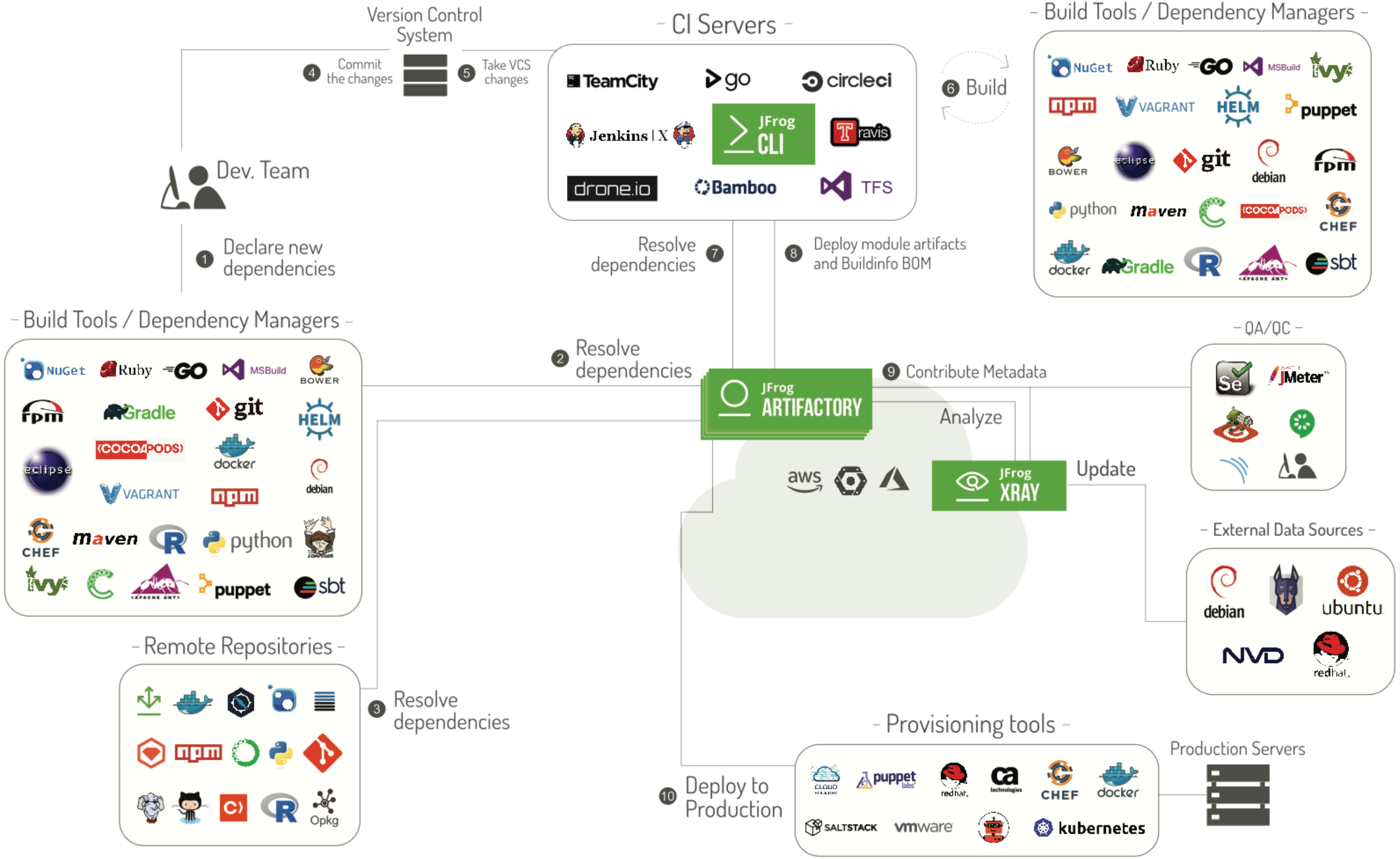

Version Control System
4 Commit the changes
5 Take VCS changes
CI Servers
TeamCity
go
circleci
Jenkins | X
JFrog CLI
Travis
drone.io
Bamboo
TFS
Dev. Team
1 Declare new dependencies
Build Tools / Dependency Managers
NuGet
Ruby
GO
MSBuild
BOWER
fpm
Gradle
git
HELM
COCOAPODS
docker
eclipse
debian
VAGRANT
npm
CHEF
maven
python
ivy
puppet
sbt
2 Resolve dependencies
JFrog ARTIFACTORY
Analyze
aws
JFrog XRAY
Update
External Data Sources
debian
ubuntu
NVD
redhat
Remote Repositories
npm
Opkg
3 Resolve dependencies

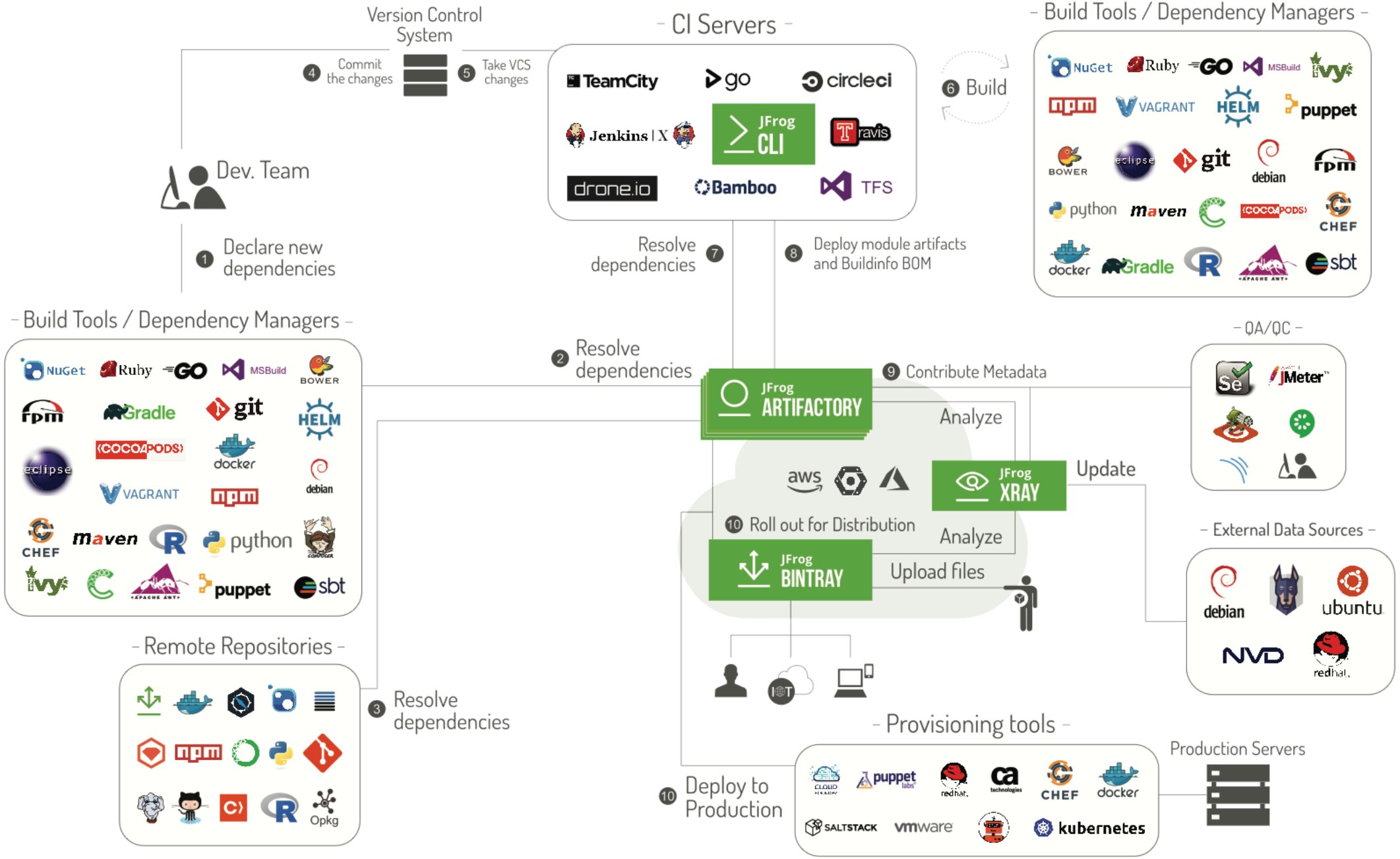

Version Control System
4 Commit the changes
5 Take VCS changes
- CI Servers -
TeamCity
go
circleci
Jenkins | X
JFrog CLI
Travis
drone.io
Bamboo
TFS
6 Build
- Build Tools / Dependency Managers -
NuGet
Ruby
GO
MSBuild
ivy
npm
VAGRANT
HELM
puppet
BOWER
eclipse
git
debian
rpm
python
maven
COCOAPODS
CHEF
docker
Gradle
R
Apache Ant
sbt
Dev. Team
1 Declare new dependencies
- Build Tools / Dependency Managers -
2 Resolve dependencies
JFrog ARTIFACTORY
Analyze
aws
JFrog XRAY
Update
- External Data Sources -
debian
ubuntu
NVD
redhat.
- Remote Repositories -
3 Resolve dependencies
Opkg

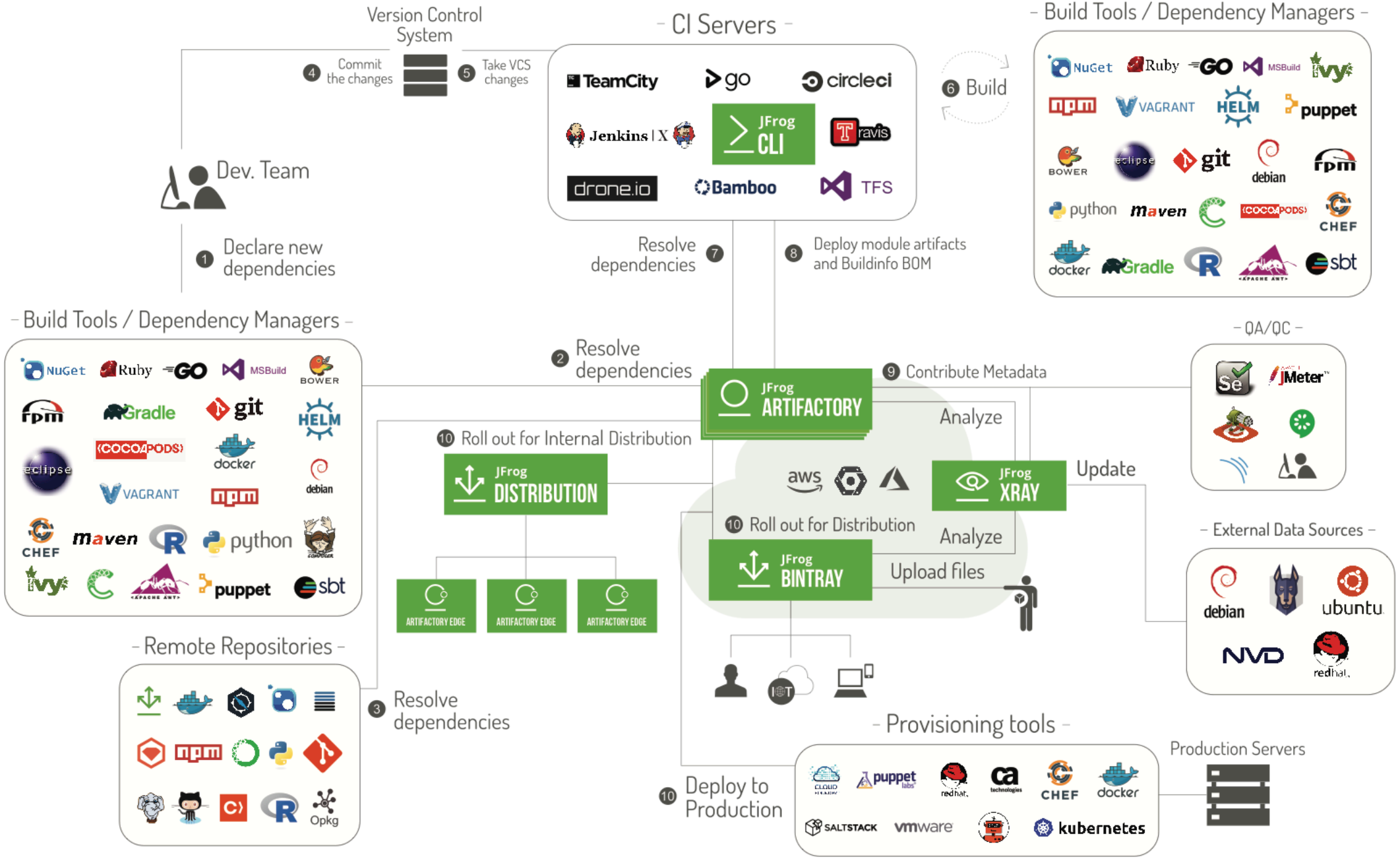

Version Control System
4 Commit the changes
5 Take VCS changes
- CI Servers -
TeamCity
go
circleci
Jenkins | X
JFrog CLI
Travis
drone.io
Bamboo
TFS
6 Build
- Build Tools / Dependency Managers -
NuGet
Ruby
GO
MSBuild
ivy
npm
VAGRANT
HELM
puppet
BOWER
eclipse
git
debian
rpm
python
maven
COCOAPODS
CHEF
docker
Gradle
R
Apache ANT
sbt
Dev. Team
1 Declare new dependencies
Resolve dependencies 7
- Build Tools / Dependency Managers -
2 Resolve dependencies
JFrog ARTIFACTORY
Analyze
aws
JFrog XRAY
Update
- External Data Sources -
debian
ubuntu
NVD
redhat.
- Remote Repositories -
3 Resolve dependencies
Opkg


Version Control System
4 Commit the changes
5 Take VCS changes
- CI Servers -
TeamCity
go
circleci
Jenkins | X
JFrog CLI
Travis
drone.io
Bamboo
TFS
6 Build
- Build Tools / Dependency Managers -
NuGet
Ruby
GO
MSBuild
ivy
npm
VAGRANT
HELM
puppet
BOWER
eclipse
git
debian
rpm
python
maven
COCOAPODS
CHEF
docker
Gradle
R
Apache Ant
sbt
Dev. Team
1 Declare new dependencies
Resolve dependencies 7
8 Deploy module artifacts and Buildinfo BOM
- Build Tools / Dependency Managers -
2 Resolve dependencies
JFrog ARTIFACTORY
Analyze
aws
JFrog XRAY
Update
- External Data Sources -
debian
ubuntu
NVD
redhat.
- Remote Repositories -
3 Resolve dependencies
Opkg


Version Control System
4 Commit the changes
5 Take VCS changes
- CI Servers -
TeamCity
go
circleci
Jenkins | X
JFrog CLI
Travis
drone.io
Bamboo
TFS
6 Build
- Build Tools / Dependency Managers -
Dev. Team
1 Declare new dependencies
Resolve dependencies 7
8 Deploy module artifacts and Buildinfo BOM
- Build Tools / Dependency Managers -
2 Resolve dependencies
JFrog ARTIFACTORY
9 Contribute Metadata
Analyze
JFrog XRAY
Update
- QA/QC -
- External Data Sources -
- Remote Repositories -
3 Resolve dependencies


Version Control System
4 Commit the changes
5 Take VCS changes
- CI Servers -
TeamCity
go
circleci
Jenkins | X
JFrog CLI
Travis
drone.io
Bamboo
TFS
6 Build
- Build Tools / Dependency Managers -
NuGet
Ruby
GO
MSBuild
ivy
npm
VAGRANT
HELM
puppet
BOWER
eclipse
git
debian
rpm
python
maven
COCOAPODS
CHEF
docker
Gradle
R
ANT
sbt
Dev. Team
1 Declare new dependencies
Resolve dependencies 7
8 Deploy module artifacts and Buildinfo BOM
- Build Tools / Dependency Managers -
2 Resolve dependencies
JFrog ARTIFACTORY
9 Contribute Metadata
Analyze
aws
JFrog XRAY
Update
- QA/QC -
Selenium
JMeter
- External Data Sources -
ubuntu
NVD
redhat
- Remote Repositories -
Opkg
3 Resolve dependencies
- Provisioning tools -
CLOUD FOUNDRY
SALTSTACK
vmware
kubernetes
10 Deploy to Production
Production Servers


Version Control System
4 Commit the changes
5 Take VCS changes
- CI Servers -
TeamCity
go
circleci
Jenkins | X
JFrog CLI
Travis
drone.io
Bamboo
TFS
6 Build
- Build Tools / Dependency Managers -
NuGet
Ruby
GO
MSBuild
ivy
npm
VAGRANT
HELM
puppet
BOWER
eclipse
git
debian
rpm
python
maven
COCOAPODS
CHEF
docker
Gradle
R
ANT
sbt
Dev. Team
1 Declare new dependencies
Resolve dependencies 7
8 Deploy module artifacts and Buildinfo BOM
- Build Tools / Dependency Managers -
2 Resolve dependencies
JFrog ARTIFACTORY
9 Contribute Metadata
- QA/QC -
Selenium
JMeter
Analyze
aws
JFrog XRAY
Update
10 Roll out for Distribution
Analyze
JFrog BINTRAY
Upload files
- External Data Sources -
debian
ubuntu
NVD
redhat
- Remote Repositories -
Opkg
3 Resolve dependencies
- Provisioning tools -
puppet labs
redhat
ca technologies
CHEF
docker
SALTSTACK
vmware
kubernetes
10 Deploy to Production
Production Servers


Version Control System
4 Commit the changes
5 Take VCS changes
- CI Servers -
TeamCity
go
circleci
Jenkins | X
JFrog CLI
Travis
drone.io
Bamboo
TFS
6 Build
- Build Tools / Dependency Managers -
NuGet
Ruby
GO
MSBuild
ivy
npm
VAGRANT
HELM
puppet
BOWER
eclipse
git
debian
rpm
python
maven
COCOAPODS
CHEF
docker
Gradle
R
ANT
sbt
Dev. Team
1 Declare new dependencies
Resolve dependencies 7
8 Deploy module artifacts and Buildinfo BOM
- Build Tools / Dependency Managers -
2 Resolve dependencies
JFrog ARTIFACTORY
9 Contribute Metadata
- QA/QC -
Selenium
JMeter
Analyze
10 Roll out for Internal Distribution
JFrog DISTRIBUTION
aws
JFrog XRAY
Update
10 Roll out for Distribution
Analyze
JFrog BINTRAY
Upload files
ARTIFACTORY EDGE
ARTIFACTORY EDGE
ARTIFACTORY EDGE
- External Data Sources -
debian
ubuntu
NVD
redhat
- Remote Repositories -
npm
Opkg
3 Resolve dependencies
- Provisioning tools -
puppet labs
redhat
ca technologies
CHEF
docker
SALTSTACK
vmware
kubernetes
10 Deploy to Production
Production Servers

DEMO TIME

Talk is cheap, show me the code!

QA AND LINKS

- www.jfrog.com/shownotes
- Slides
- Video (tomorrow!)
- All the links!
- Ratings, comments
- Raffle!