Facilitating continuous delivery in a FinTech world with Salt, Jenkins, Nexus and Chocolatey

Michel Buczynski, DevOps Coach
TD Securities

Disclaimer

While this session is mainly based on work done for various employers and my contemporary observations, this session will focus on my personal opinions. As such, material herein is not necessarily representative of my present or my previous employers' opinion.

Who am I

Born in '73 and raised in Montreal, work and live in Toronto.

About me:

- I disassembled (hum! Destroy) the family TV at 3 years old to understand where the images and sounds were coming from.
- I got my first programming contract at 16.
- I practiced Agile before Agile, and DevOps before DevOps, and I think I am not alone.

What I am doing now

DevOps Coach

- Implementation of a tools-chain for Continuous Delivery (CD).
- Define a standardized workflow for the CD pipeline.
- Coach team members on how to properly use the tools-chain and follow the workflow.
- My goal is to ensure that our team produces, delivers, and maintains efficient and quality software in the easiest and fastest possible manner.

Current Gig

TD Securities is part of the TD Bank Group

- The Toronto-Dominion Bank & its subsidiaries are collectively known as TD Bank Group (TD). TD is the sixth largest bank in North America by branches & serves approximately 22 million customers in a number of locations in key financial centers around the globe. Over 85,000 TD employees represent the strongest team in banking. Delivering legendary customer experiences is who we are & is part of being the Better Bank.
- I am working in the capital markets division (TD Securities); specifically for the global equity derivatives business.

FinTech

- FinTech Wikipedia definition:
  Financial technology (FinTech or fintech) is the new technology and innovation that aims to compete with traditional financial methods in the delivery of financial services.
- FinTech uses the latest IT technologies like Big Data, IoT, Machine Learning, etc.
- The banking industry is the major provider of financial services.
- Since the banking industry was one the first users of computer systems, banks sometimes have to rely on legacy systems.

The Challenge of DevOps in the Banking Industry

- The size and the age of the enterprise.
- Regulations, compliance, audits, separation of duties, …
- A varied infrastructure based on both legacy and modern technologies.
- The use of 3rd party and legacy software.
- New Technology Introduction (NTI).
- Conservative approach with a very strong need for stability.
- Changes will be always difficult to apply because it is especially difficult to change the way people works. But, it is also difficult to make change without changing the way people work.

What our Team Does

- We are a cross-functional team dedicated to producing cutting-edge FinTech software.
- Most of our development is based on FOSS (Free Open Source Software) framework or platform: Node.JS, Erlang/Elixir, Angular5, Scala, Python, R and Java.
- Our software architecture is microservice oriented.
- Our system is integrated with some legacy COTS (Commercial Off-The-Shelf) applications tied to Windows desktop and server.
- Developing in-house software permits us to have more flexible and adapted software, effectively leveraging our knowledge capital and reducing our dependency on 3rd parties.

Our Team's DevOps Journey

A Long Journey Begins with a Single Step

- Most of our application code uses version control (VC) GIT.
- No centralized GIT VC systems employed.
- No automated testing.
- Weekly painful manual deployment. It was honestly an achievement to have such a short yet manual release cycle.

Our Team's DevOps Journey

Today

- Application code, configuration (source) and packaged artifacts (target) are all in VC.
- Continuous Integration.
- Automated deployments.
- Most of our system configuration is automated.
- Most of the application testing is automated.

Our Team's DevOps Journey

The Future

- Provide self-serve services and resources for non-developers. Ex: Release management, pick commit for release candidate, release notes automation, etc…
- Make automated testing of infrastructure changes.
- Make provisioning automated on all targets.
- Make success metrics visible.
- Make incident responses automated.

Standardize your Workflow

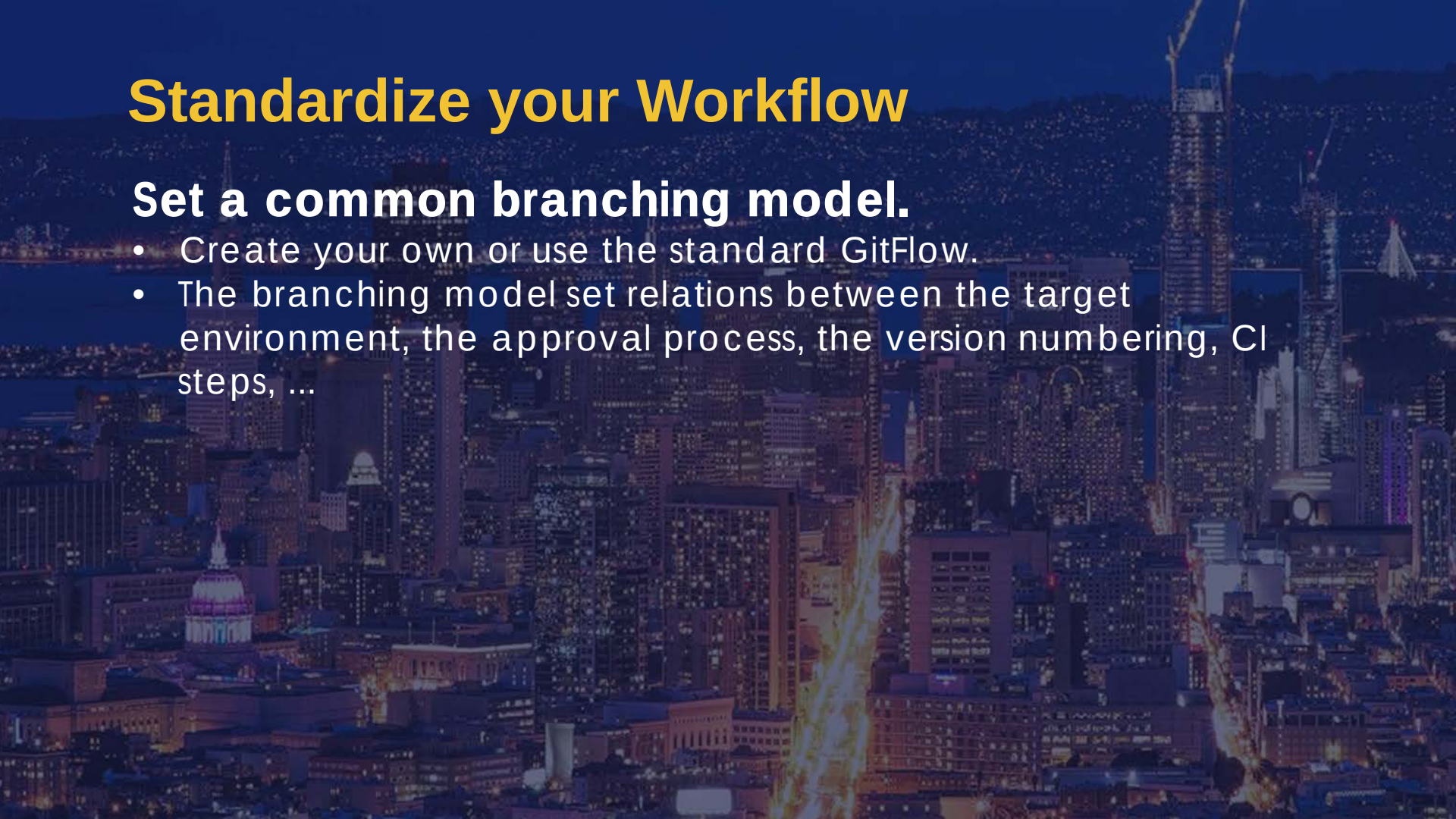
Everything in VC

- Source code, Source library, Application configs.
- Artifacts, System configuration.
- Package everything, put it in a VC repositories. Especially what is not your own code.
- The good, the bad and the ugly about .MSI.
- Chocolatey: the solution for Windows.

Standardize your Workflow

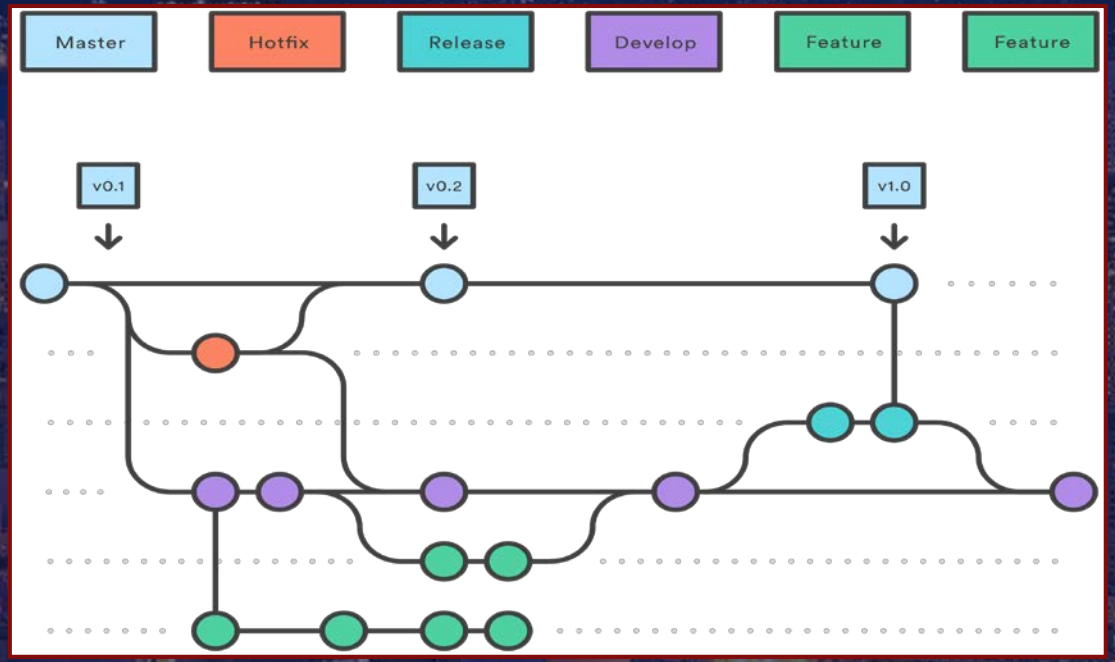
Set a common branching model.

- Create your own or use the standard GitFlow.
- The branching model set relations between the target environment, the approval process, the version numbering, CI steps, ...



Standardize your Workflow

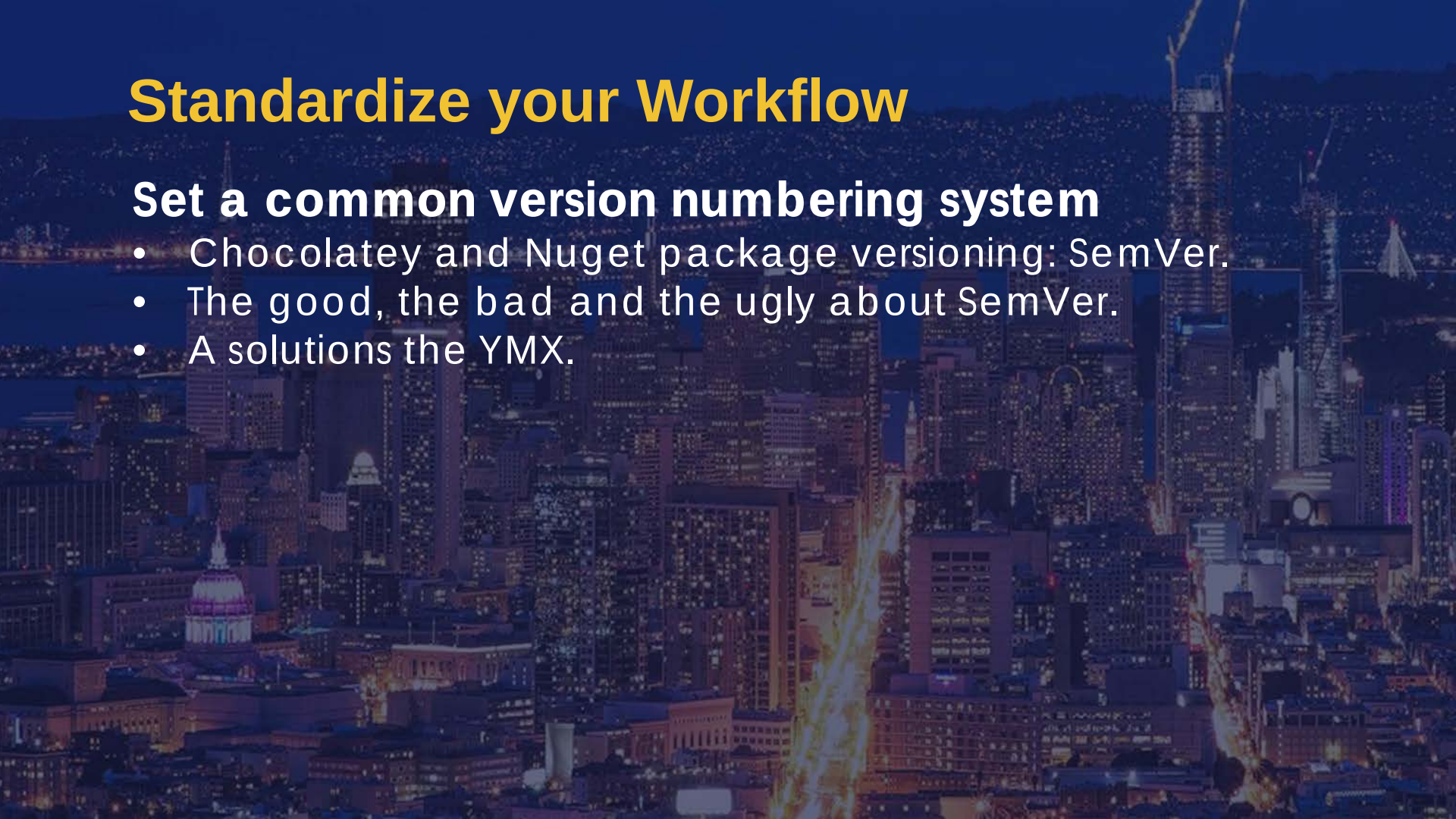
GitFlow



Standardize your Workflow

Set a common version numbering system

- Chocolatey and Nuget package versioning: SemVer.
- The good, the bad and the ugly about SemVer.
- A solutions the YMX.



Standardize your Workflow

Chocolatey and Nuget package versioning.

All version Chocolatey and Nuget support SemVer 1.0 :
A specific version number is in the form *Major.Minor.Patch[-Suffix]*, where the components have the following meanings:

- *Major*: Breaking changes
- *Minor*: New features, but backwards compatible
- *Patch*: Backwards compatible bug fixes only
- *-Suffix* (optional): a hyphen followed by a string denoting a pre-release version.

Examples:

```
1.0.1
6.11.1231
4.3.1-rc
2.2.44-beta
11.0.1-alpha
```

All version Chocolatey and Nuget also support Microsoft Version Numbers:
A specific version number has the a 4 number form: *Major.Minor.Buid.Revision*
Chocolatey call this form : *Package Fix version Notation*
Examples: `1.2.0.20181008`

Standardize your Workflow

The good, the bad and the ugly about SemVer.

The good:

- Has release and pre-release version
- Tell you if you need to update:
  - MAJOR: Breaking API change
    Not safe to update
  - MINOR: New features
    Safe to update
  - PATCH: Bugfixes
    Must update

Standardize your Workflow

The good, the bad and the ugly about SemVer.

The bad:

- With rapid space of CD is difficult to follow MAJOR, MINOR and PATCH ordering.
- It's assumes that every goes well and arrive on time.
- It's only great once the product is done and tested.

Standardize your Workflow

The good, the bad and the ugly about SemVer.

The ugly:

- Difficult to automate.
- Need a lots manual intervention.
- With things like versioning of things API, the MAJOR and MINOR doesn't mean anything for artifacts version.
- You can always put the major version in the product name. Ex: X11, Oracle12c and C

RELEASE CANDIDATE Version for release/ and hotfixe/ branch:
YY.M.N-rcX

Standardize your workflow

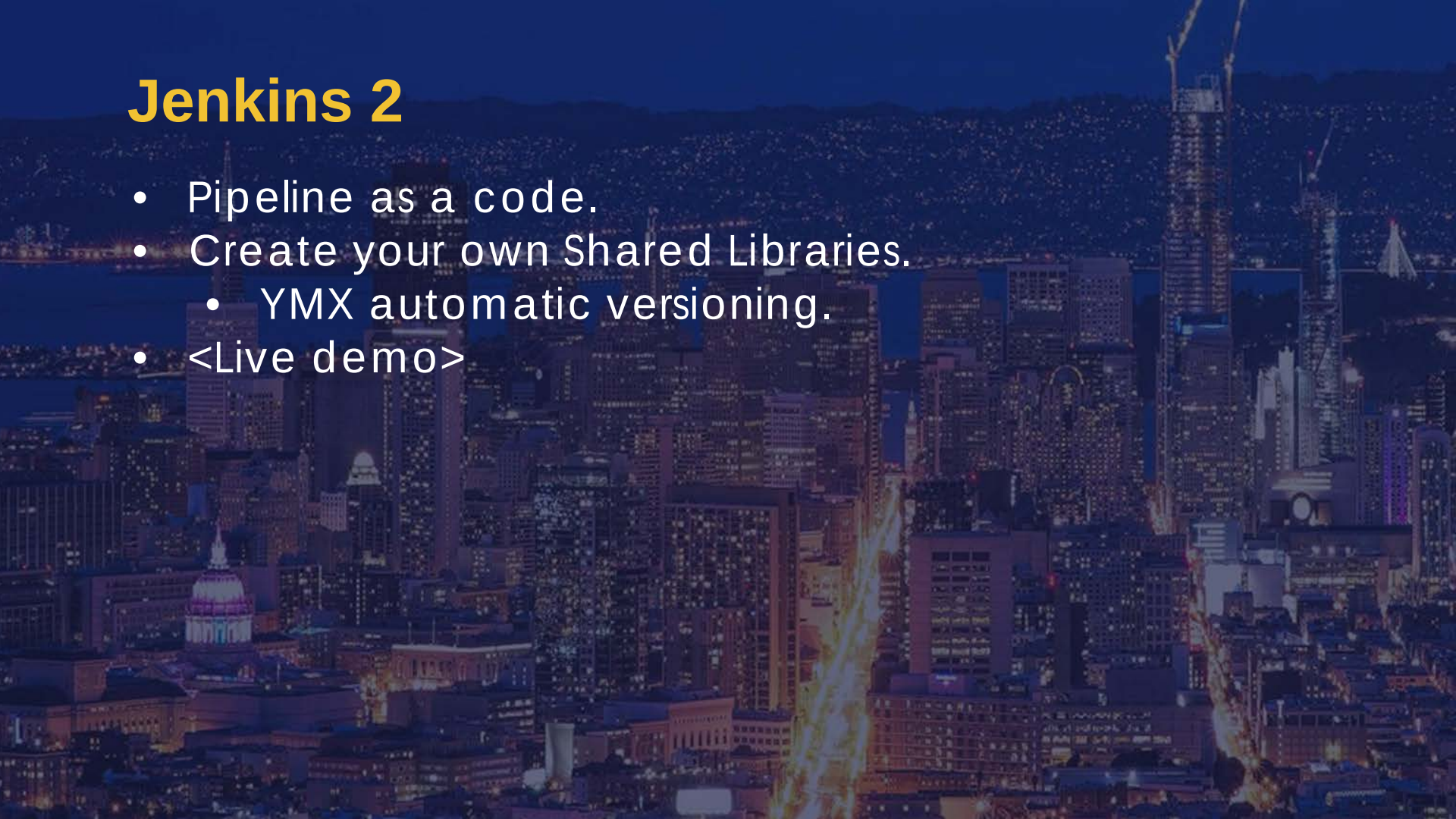
A solution the YMX.

- Inspired on Docker's versioning scheme
- **RELEASE version for *master* branch: YY.M.N**
  18.1.9
- **RELEASE CANDIDATE Version for release/ and hotfixe/ branch: YY.M.N-rcX**
  18.1.9-rc 18.3.2-rc4
- **ALPHA Version for feature/ branch: YY.M.N-alpha-DD-hhhhhhh**
  18.3.3-alpha-08-df81230 for the commit done the Mars 8th 2018 in preparation for third release in Mars 2018

Standardize your Workflow

Clean up and Enforce

- Clean up all your repositories.
- By doing a clean-up, it is the right time to introduce new tools and processes.
- Train your team to use new tools and teach them how to apply your workflow and demonstrate why.
- Enforce workflow progressively by applying it, with a small group of team members and target applications.



Jenkins 2

- Pipeline as a code.
- Create your own Shared Libraries.
  - YMX automatic versioning.
- <Live demo>

Jenkins 2

Pipeline as a code.

- *Pipeline as Code* describes a set of features that allow Jenkins users to define pipelined job processes with code, stored and versioned in a source repository. These features allow Jenkins to discover, manage, and run jobs for multiple source repositories and branches — eliminating the need for manual job creation and management.
- To use Pipeline as Code, projects must contain a file named *Jenkinsfile* in the repository root, which contains a "Pipeline script."

Jenkins 2

Create your own Shared Libraries.

- As Pipeline is adopted for more and more projects in an organization, common patterns are likely to emerge. Oftentimes it is useful to share parts of Pipelines between various projects to reduce redundancies and keep code "DRY".
- Pipeline has support for creating "Shared Libraries" which can be defined in external source control repositories and loaded into existing Pipelines

Jenkins 2

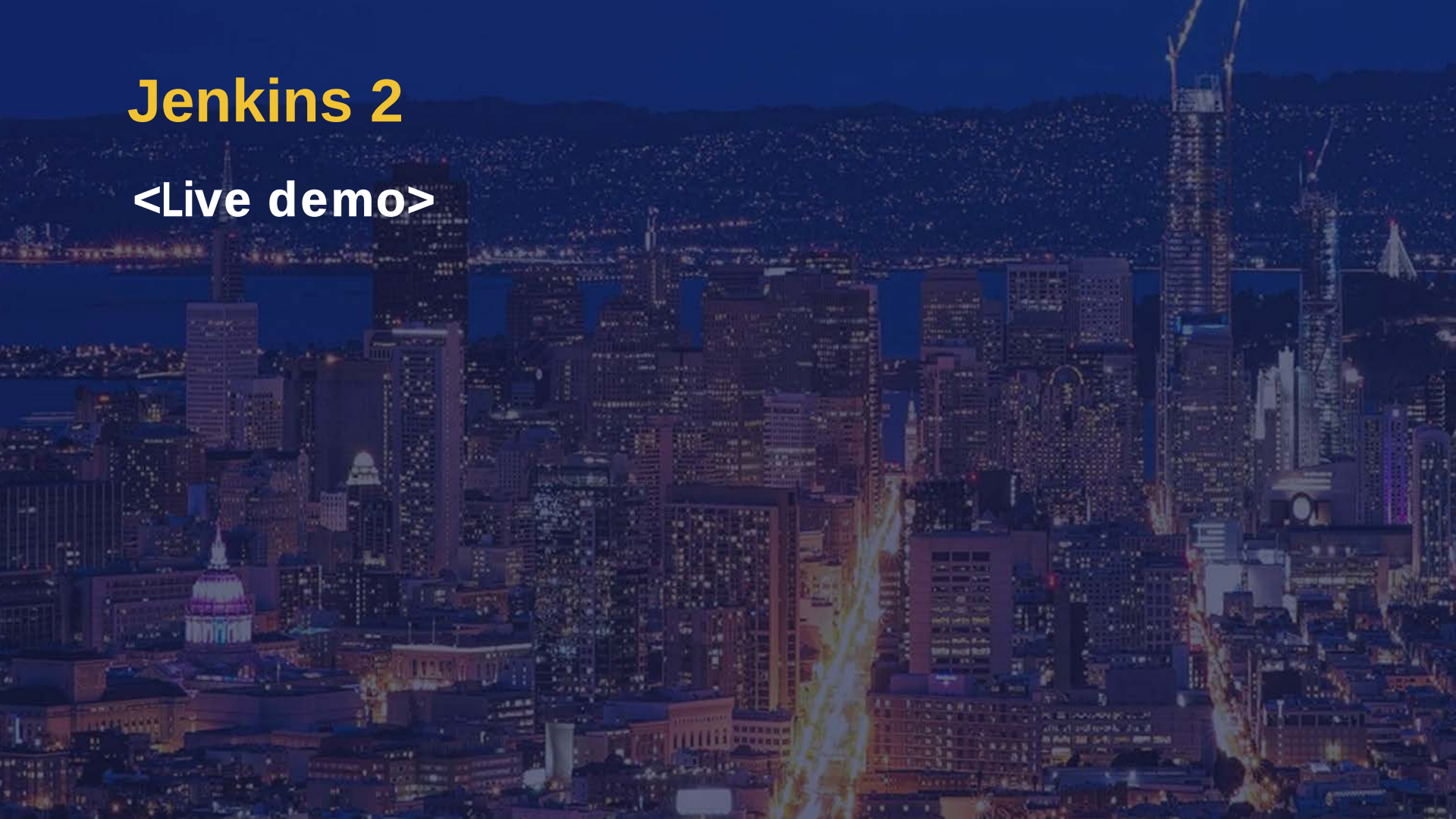
Create your own Shared Libraries.

- Declarative Pipeline
- Scripted Pipeline

Directory structure

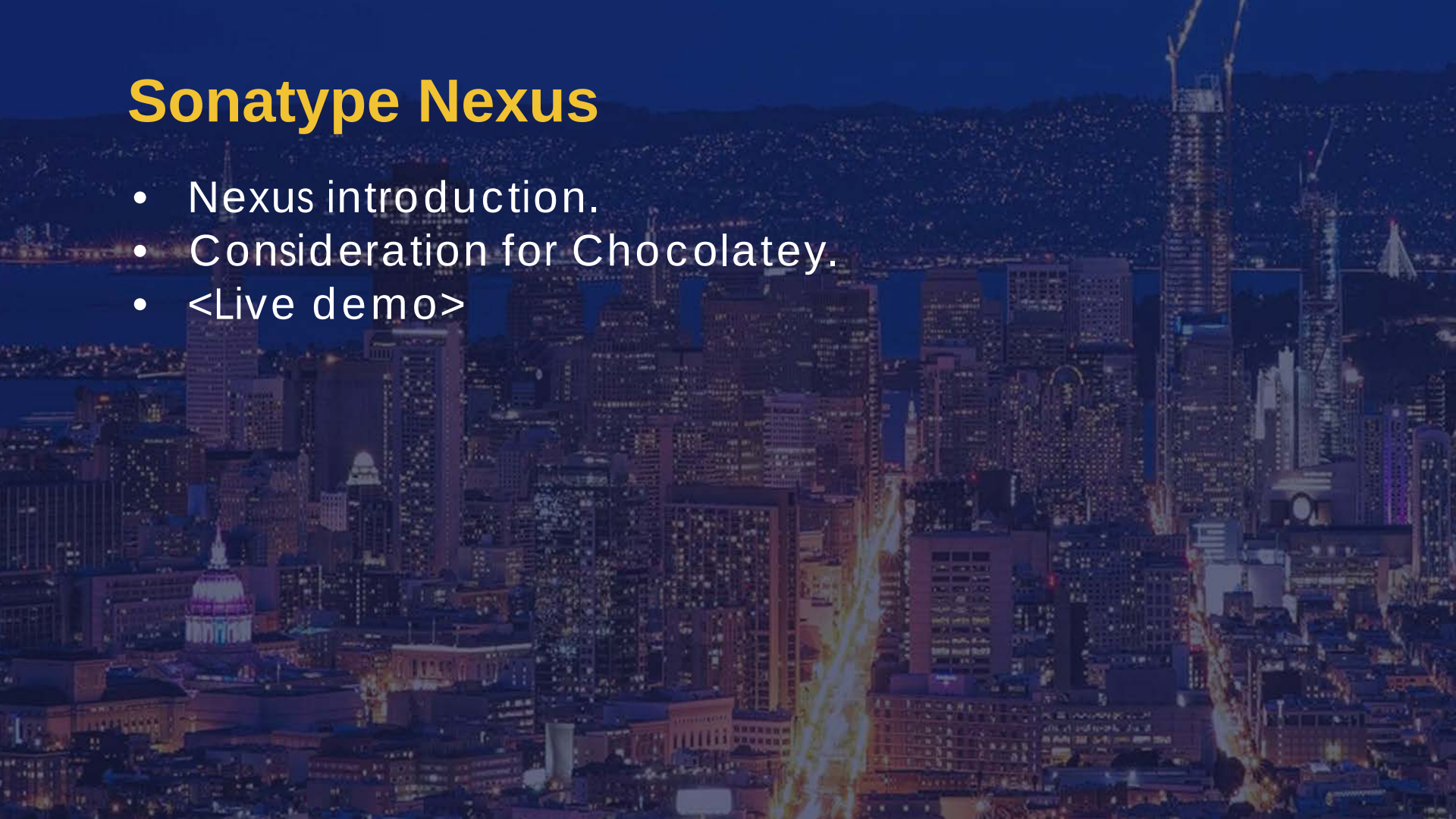
The directory structure of a Shared Library repository is as follows:

```
(root)
 +- src                     # Groovy source files
 | +- org
 |      +- foo
 |          +- Bar.groovy   # for org.foo.Bar class
 +- vars
 |   +- foo.groovy          # for global 'foo' variable
 |   +- foo.txt             # help for 'foo' variable
 |   +- bar.groovy          # bar custom step, call def call()
 +- resources               # resource files (external libraries only)
 |   +- org
 |        +- foo
 |            +- bar.json   # static helper data for org.foo.Bar
```


Jenkins 2
<Live demo>

Sonatype Nexus

- Nexus introduction.
- Consideration for Chocolatey.
- <Live demo>



Sonatype Nexus

Nexus Reposory Manager introduction

- Universal package manager

  That give us the ability to apply security and compliance metrics across all artifact types. Universal package managers have been referred to as being at the center of a DevOps toolchain

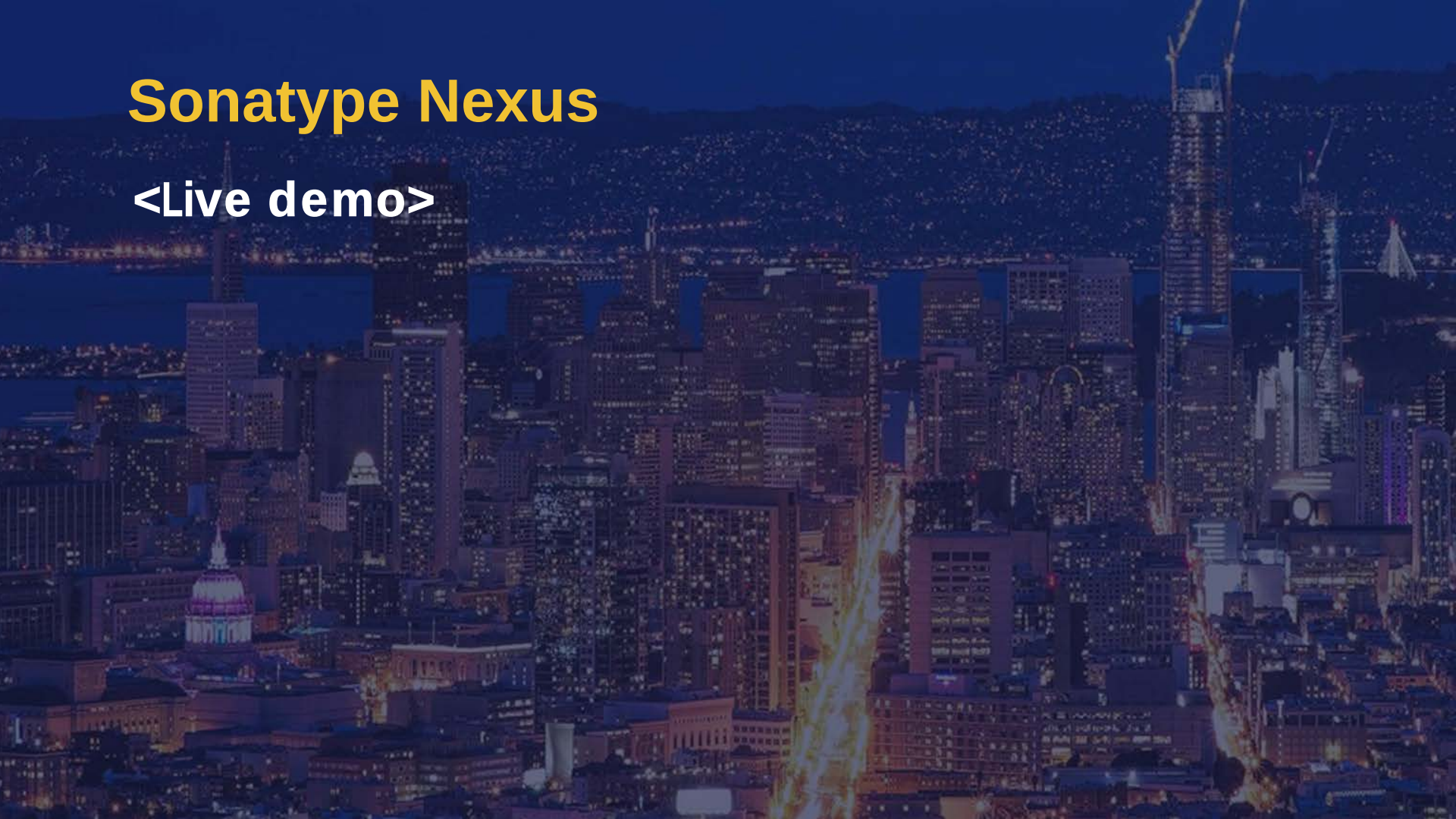
- Nexus Repository OSS 3.xx, supports those formats:

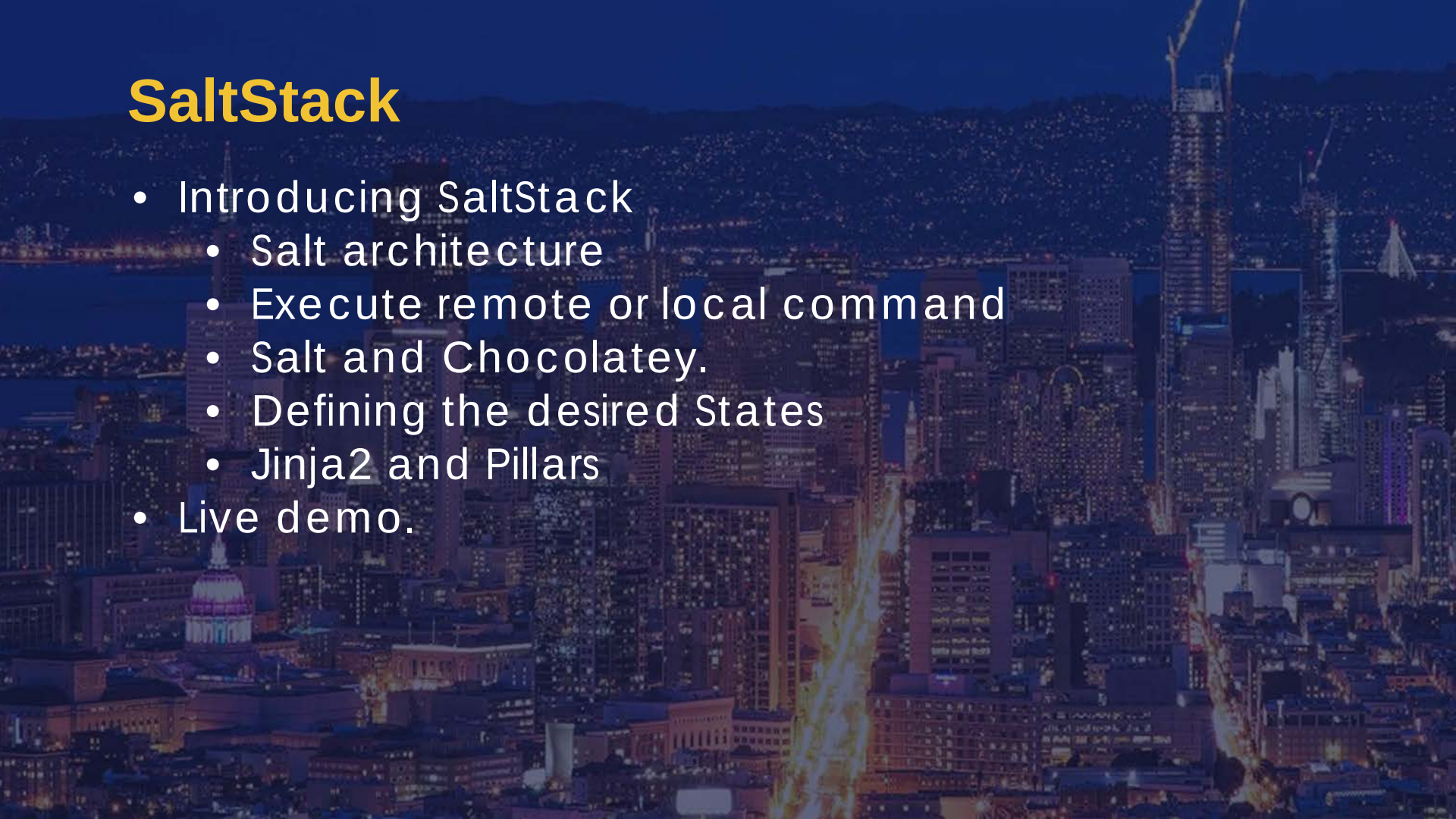
  APT, Composer, Conan, CPAN, Docker, ELPA, Git LFS, Helm, Maven, npm, NuGet, P2, PyPI, R, Raw, RubyGems, Yum

Sonatype Nexus

Consideration for Chocolatey.

- Don’t mix Nuget package with Chocolatey package. Even if Chocolatey is based on the NuGet framework, it doesn’t mean that they are the same type of package.
- Separated in different repositories, depending on the target. Users workstations, developer workstations, agents workstations, servers. You can also create separate repositories for certain products.


Sonatype Nexus
<Live demo>



SaltStack

- Introducing SaltStack
  - Salt architecture
  - Execute remote or local command
  - Salt and Chocolatey.
  - Defining the desired States
  - Jinja2 and Pillars
- Live demo.

SaltStack

Introducing SaltStack

- Open-source configuration management software and remote execution engine.
- Infrastructure as code
- Python based
- Use asynchronous messaging queue
- Fast and scalable
- Mainly referred as “Salt”

SaltStack

Salt architecture

- Designed for high speed data collection and execution in system administration environments. At beginning Salt was relying on ZeroMQ. Salt now has is own realiable queuing transport system: RAET (Reliable Asynchronous Event Transport Protocol). Which permit Salt to be scalable well beyond tens of thousands of servers.

SaltStack

Salt architecture

- Salt mainly use a slave-master setup, that enables Salt to do push or pull remote execution. The slave or the agent is called : Minion.
- Minion can be also be used alone in a MasterLess mode.
- Salt has also AgentLess mode called: Salt-SSH. Salt has no Windows AgentLess mode yet.

SaltStack

Salt architecture

- Salt mainly use a slave-master setup, that enables Salt to do push or pull execution. The slave or the agent is called : Minion.
- Minion can be also be used alone in a MasterLess mode.
- Salt has also AgentLess mode called: Salt-SSH. Salt has no Windows AgentLess mode yet.
- Salt permit also event-driven execution and self-healing.

SaltStack

Salt architecture

- The modular design of Salt is done by module written in Python. By the abilities to write your own Salt Module, Salt is easily extensible.
- Module types:
  - **Execution modules**
  - **State modules**
  - **Grains**
  - **Renderer modules**
  - **Returners**
  - **Runners**

SaltStack

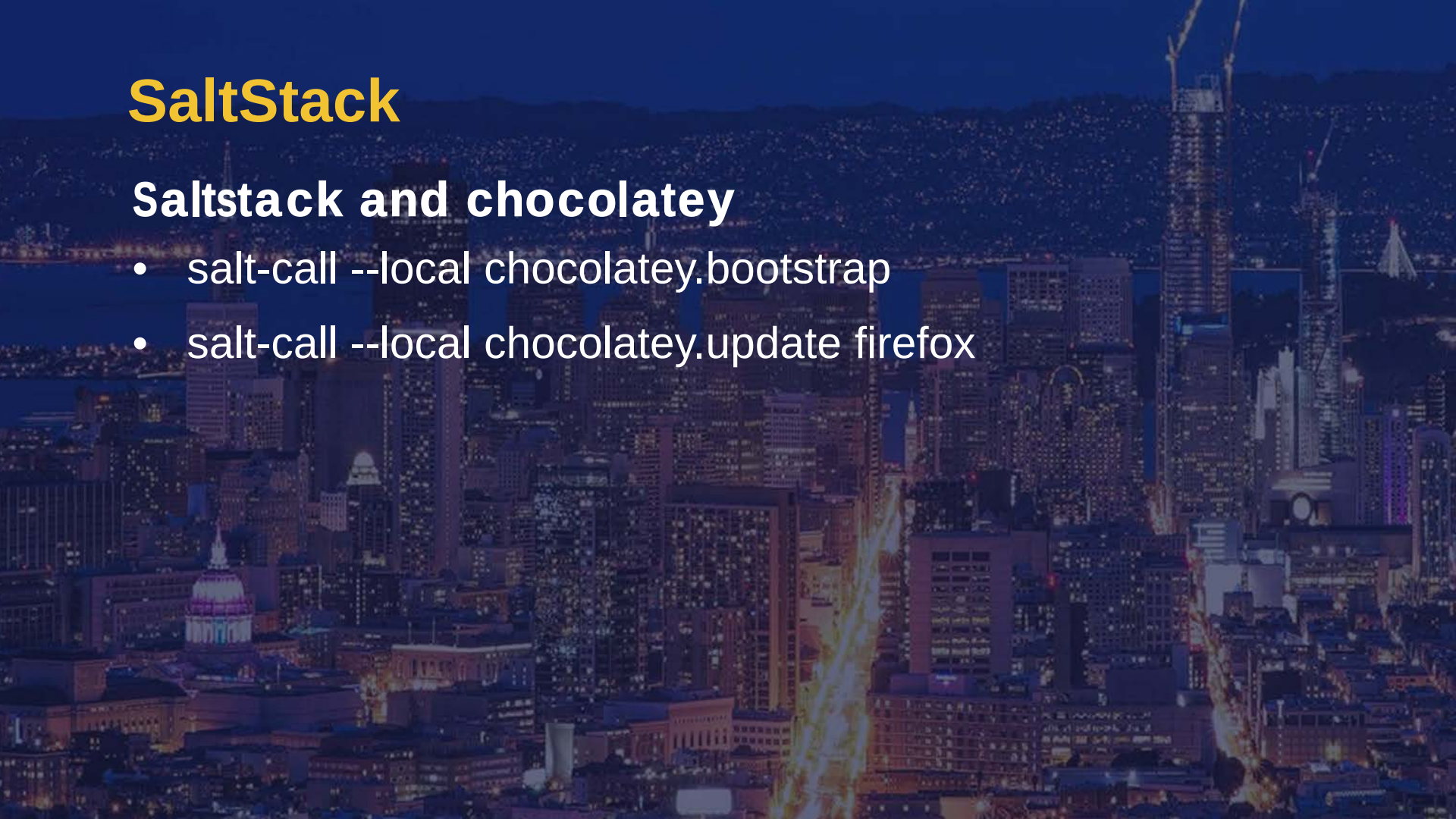
Execute remote command

- salt-call '*' test.ping
- salt-call '*' disk.usage
- salt-call -G 'os:Windows' cmd.run ‘dir‘
- salt-call –local cmd.powershell “Get-ChildItem c:\”

SaltStack

Saltstack and chocolatey

- salt-call --local chocolatey.bootstrap
- salt-call --local chocolatey.update firefox



SaltStack

Defining the desired States

- firefox.sls :

```
chocolatey:
  pkg.installed
firefox:
  chocolatey.upgraded:
    - pkgs:
      - firefox
```

- salt-call state.apply firefox

SaltStack

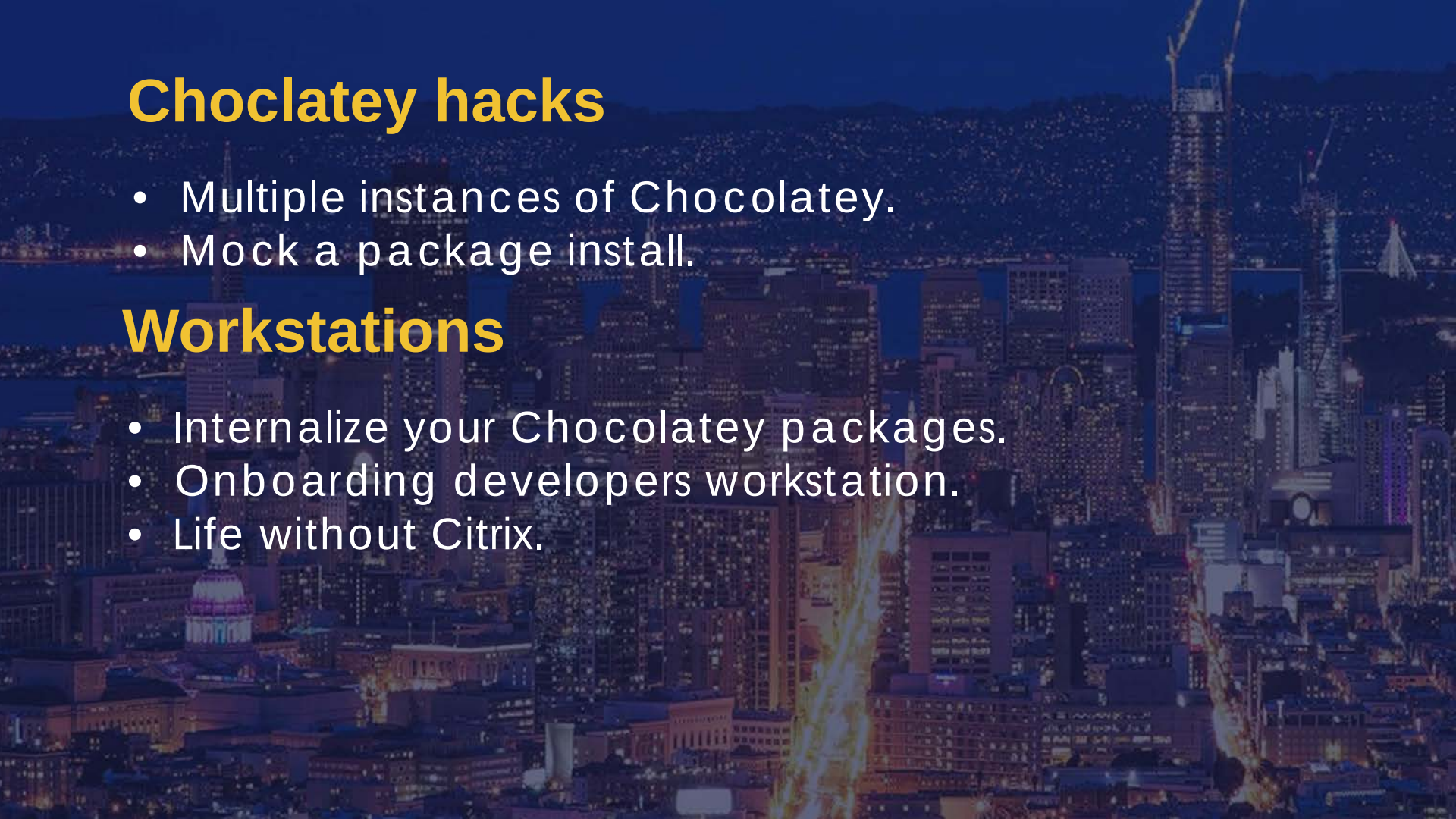
Jinja2 and Pillars

- Using grains in SLS

```
apache:
{% if grains['os'] == 'RedHat' %}
  pkg.installed:
      - name: httpd
{% elif grains['os'] == 'Ubuntu' %}
  pkg.installed:
      - name: apache2
 {% if grains['os'] == ‘Windows' %}
  chocolatey.installed:
     - name: apache-httpd
 {% endif %}
```

- Pillars


SaltStack
<Live demo>



Choclatey hacks

- Multiple instances of Chocolatey.
- Mock a package install.

Workstations

- Internalize your Chocolatey packages.
- Onboarding developers workstation.
- Life without Citrix.

What we've Learned

- Patience. Renovating a house will always take more resources than building a new one. But the importance is to always be evolving.
- Impose standardization of your process. Less experienced team members will often have a certain lack of discipline. But some older team members are more resistant to change their discipline.
- Make the jobs fun. Try to replace boring tasks by automated process or try to reduce time past on those boring tasks.

What we learn

- Try to avoid everything that is not human readable like .MSI, .XML and Windows registry.
- Click, click alone is really very bad. If you can write a how-to wiki. You can write a script (preferably with a CM). If you have a good script, you can do a one-liner command. With that you can easily create a WebUI or GUI for a secure self-service.
- Concentrate on useful metrics and make only useful alerts.
- Innovate and don't be afraid to push the envelope

Thanks

- A special thanks to my team at TD.
- To Rob Reynolds, to make me love Windows again.
- Chocolatey Fest team…

Q&A