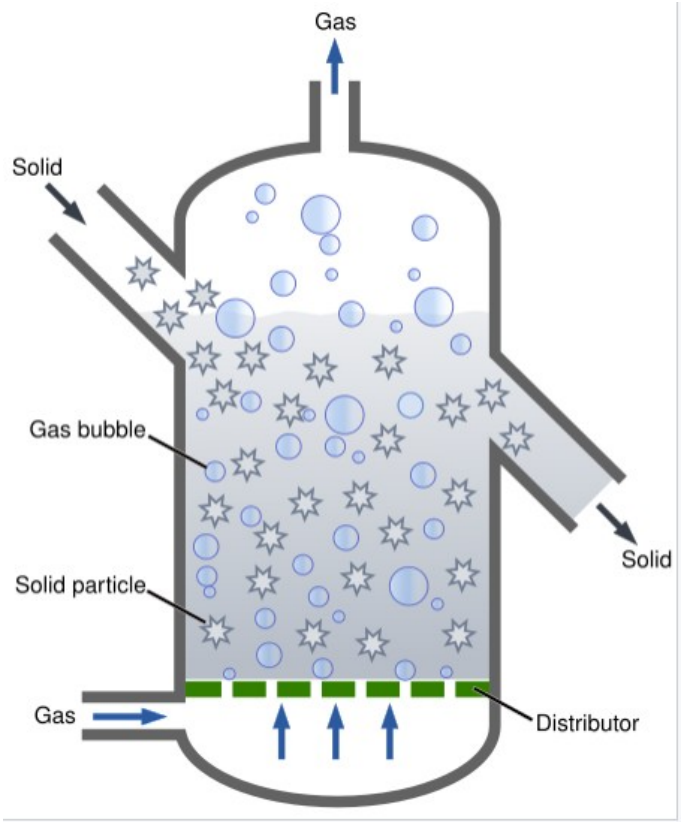
PWP Fluidization Overview



Typical Enterprise App Install

- **Configuration Management** (**CM**) **Tool** installs Runtimes according to hierarchy configurations, system wide – the **CM Tool** effectively *is* Configurations **hierarchy** + Operations **DAG**
- **CM Tool** triggers **App** install
- **App** Installer unpacks the app package
- Embedded runtime is ignored / pruned
- At execution time no custom step is required to find embedded **JDK -** it is *hard coded*

Custom PWP Install Steps

A number of actions is delegated to the Installer:

- unpacks the app package with an **embedded JDK**
- Legacy version had **embedded JDK** path hard coded *at multiple levels*
- New version inspects if **Enterprise JDK** is present (directory scan, may vary with how visible the **Enterprise JDK** installs itself)
- creates a `custom.properties` file with **JDK info**
- Inside the app, every JDK aware code and scripts is updated to utilize `custom.properties`

Another Runtime hosting Tool Example

Puppet Testing Framework - similar to (predecessor of) **Chef Inspec**

- **CM Tool** installs **app** which installs **Ruby Runtime** environment in custom location
- **Ruby**, **Python**, **Java**, **Node.JS**, **Go**, **.Net** natively support multiple **side by side** environments.
- **Ruby** is the origin of **ServerSpec** (replicas exist). There is no system Ruby by default. **PTF** relies on **uru** (kind of **rvm**) and **Puppet** for metadata interpolation
- Integration metrics collected at specific **Puppet** stage
- App and its embedded Runtime is removed by **CM Tool**

Configuration

Minimal override hierarchy

1) file from `$(user.dir}`
2) resource from the assembly jar
3) hard coded in the source
4) specific location when packed as Maven plugin

Minimal String Interpolation – optional, handy

```
Foo: 42
Bar: answer is ${foo}
Home: ${USERPROFILE}
```

Can handle formats: `YAML,INI,JSON,XML`