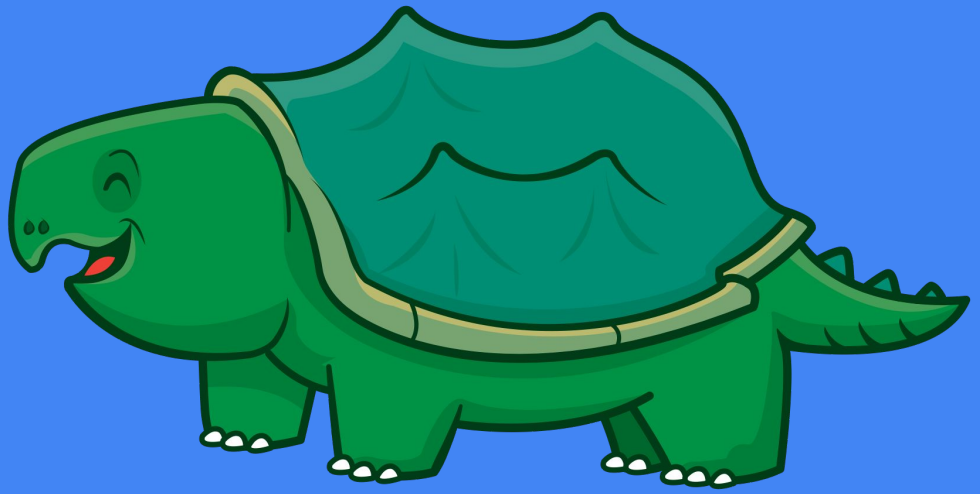
Snap

A story of telemetry, data, python
And references to Jason Dixon at
GrafanaCon 2016

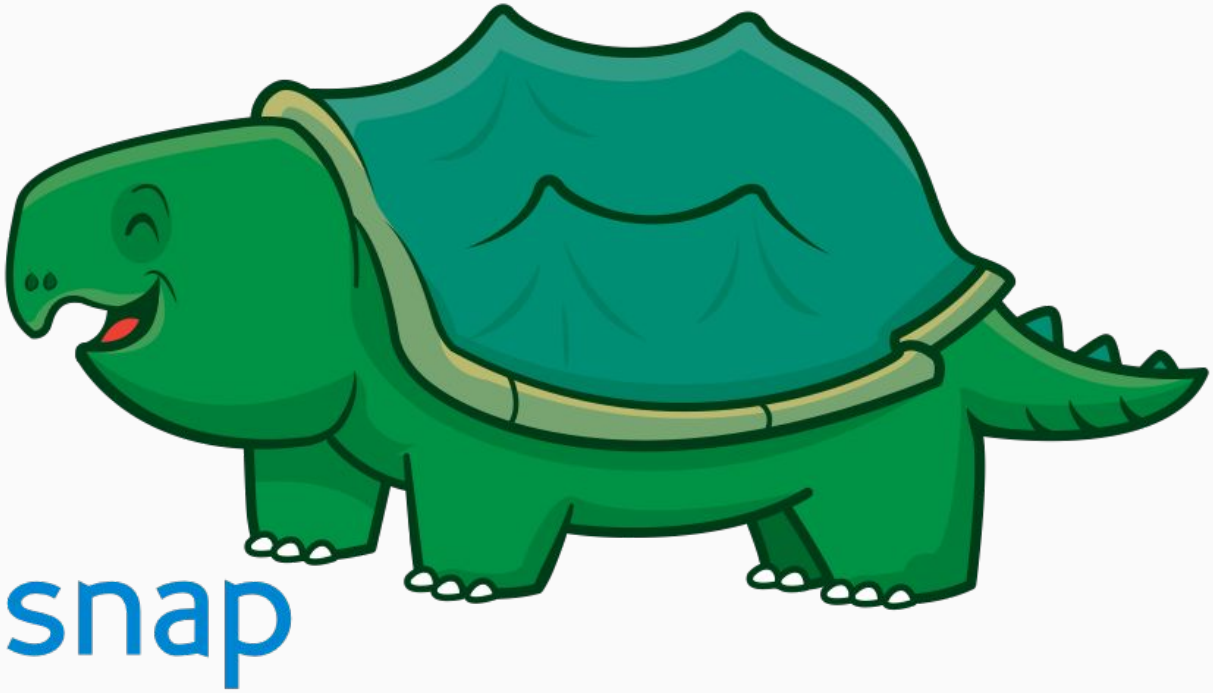


Hey

MATTHEW BRENDER

JOEL COOKLIN

The Snap Telemetry Framework

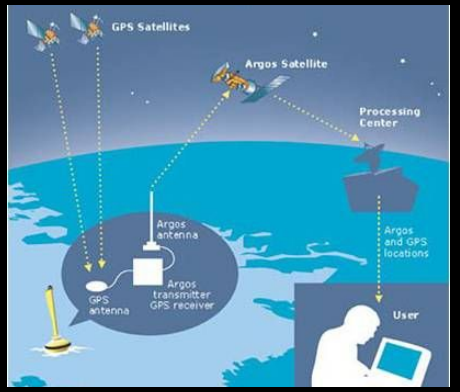


the open telemetry framework

snap-telemetry.io

what my friends think telemetry is

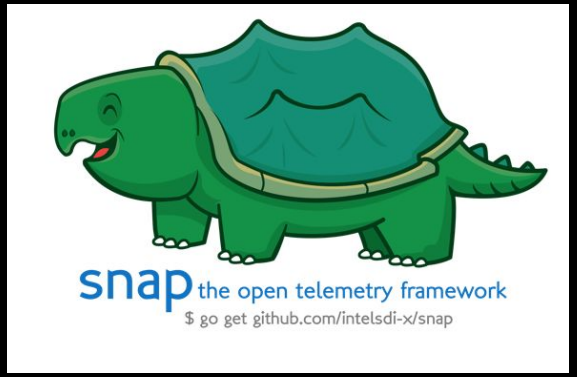
what my parents think telemetry is



what society thinks telemetry is

what my boss thinks telemetry is

what I think telemetry is



what telemetry actually is

“Telemetry refers to the collection of measurements, typically of remote instruments, for the purposes of monitoring.”

~ Jason Dixon in The Graphite Book

(Do I get bonus points for quoting Jason at GrafanaCon?)

Telemetry Framework ™

- Snap
- collectd
- telegraf
- diamond
- beats
- tcollector
- scollector
- Nagios

Client libraries for:

- Prometheus
- Sensu
- Statsd
- & more

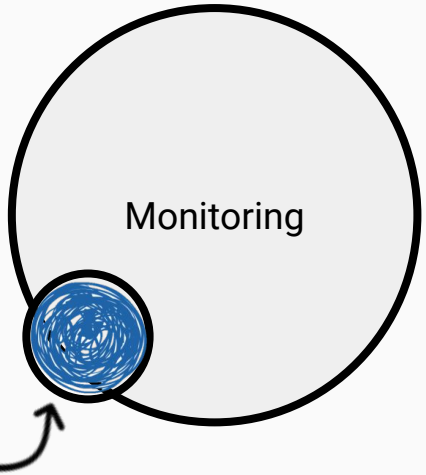
Requirements

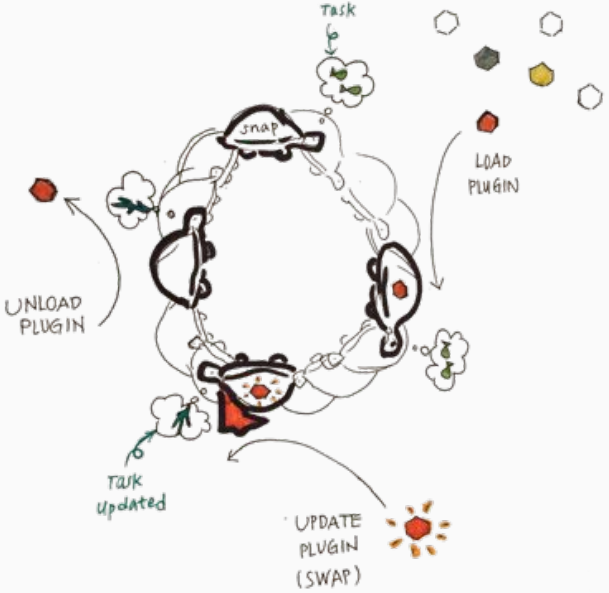
Composable

Declarative

Extensible

Composable


Monitoring
Snap


Task
Snap
LOAD PLUGIN
UNLOAD PLUGIN
Task Updated
UPDATE PLUGIN (SWAP)

Composable

Monito

Snap

MySQL

cassandra

kafka

PostgreSQL

RabbitMQ

intel

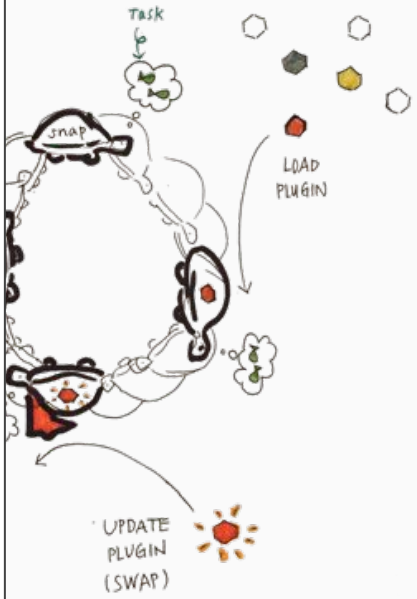
cAdvisor

openstack
CLOUD SOFTWARE

elastic

OpenTSDB

influxdata



Declarative

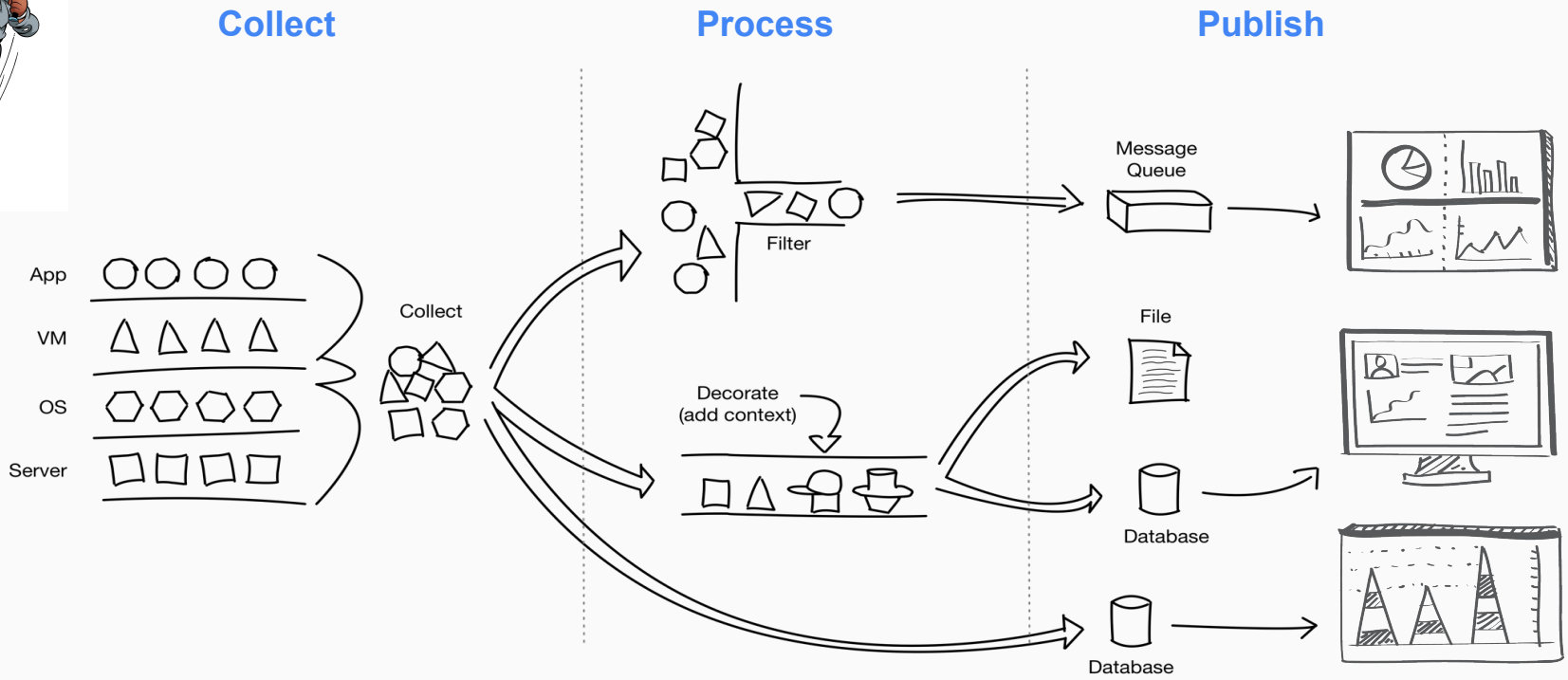
“I’ll make you a workflow you can’t refuse”

Explicit definition of
- Interval
- Collected metrics
- Published endpoints

Allows for
- Collect from multiple sensors
- Publish to multiple endpoints
- Multiple workflows at different intervals
- Metrics available upon plugin update

Let’s look at a **Task Manifest**

Extensible


Collect
Process
Publish
App
VM
OS
Server
Collect
Filter
Message Queue
Decorate (add context)
File
Database
Database

Extensible

STAPLES
Make More Happen

Collectors written:

- Nginx
- Couchbase
- MongoDB
- Netstat
- Procstat

Publishers written:

- Blueflood
- Couchbase

Available in the Plugin Catalog

Extensible

STAPLES
Make More Happen

Collectors written:

- Nginx
- Couchbase
- MongoDB
- Netstat
- Procstat

Publishers written:

- Blueflood
- Couchbase

Available in the [Plugin Catalog]

NETFLIX

Extensible

STAPLES
Make More Happen

Collectors written:

- Nginx
- Couchbase
- MongoDB
- Netstat
- Procstat

Publishers written:

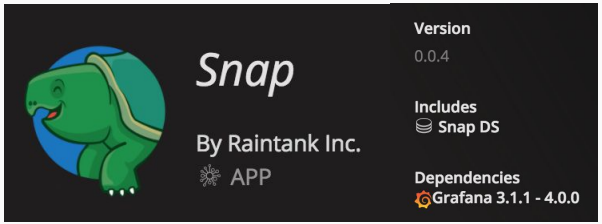
- Blueflood
- Couchbase

Available in the Plugin Catalog

NETFLIX

docker → Atlas

raintank



Extensible

STAPLES
Make More Happen

Collectors written:

- Nginx
- Couchbase
- MongoDB
- Netstat
- Procstat

Publishers written:

- Blueflood
- Couchbase

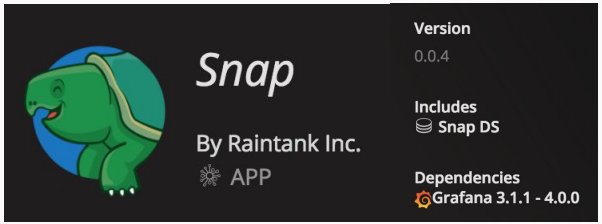
Available in the Plugin Catalog

NETFLIX

docker → Atlas

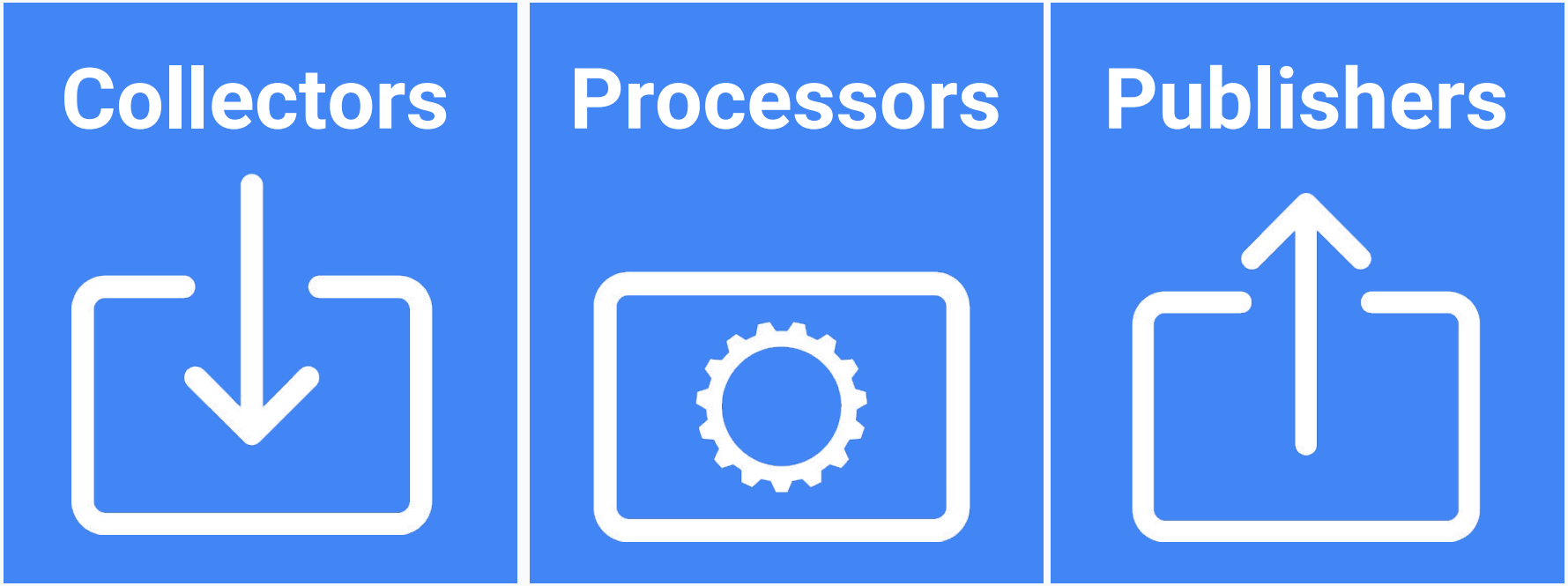
Grafana Labs

raintank

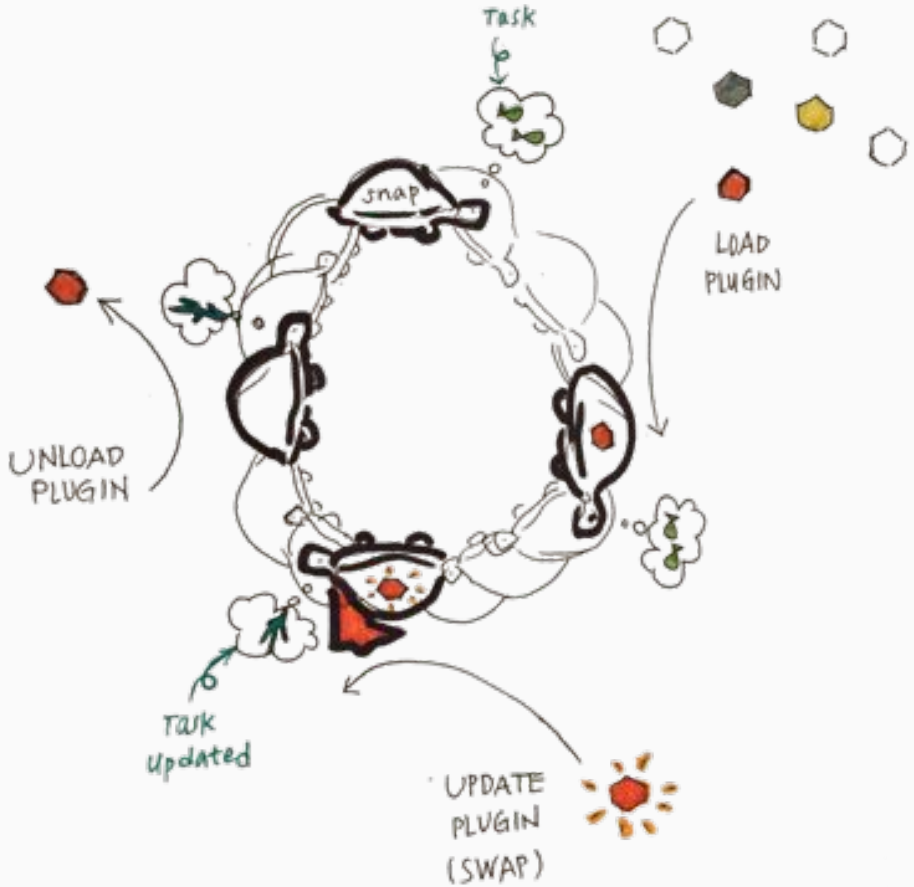


The Anatomy of a Snap Plugin

Plugin Authoring - Plugin Types



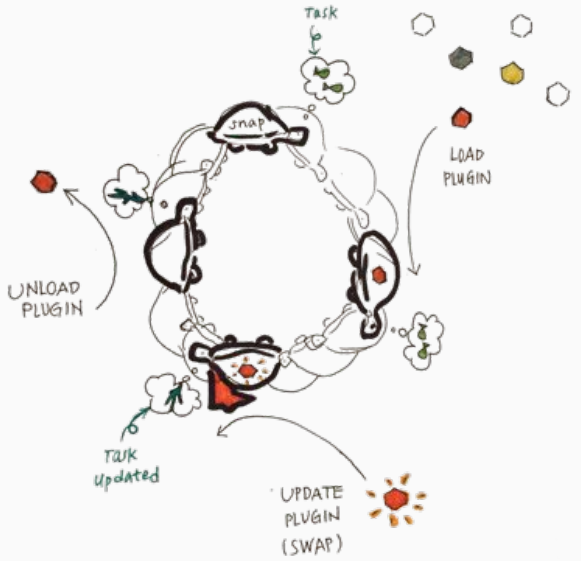
Loading a plugin



Plugin Authoring - Lifecycle

Loading a plugin

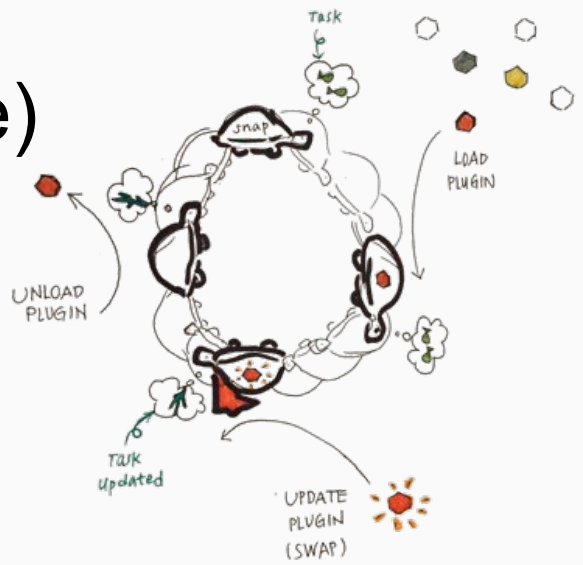
1. Snap starts the plugin



Plugin Authoring - Lifecycle

Loading a plugin

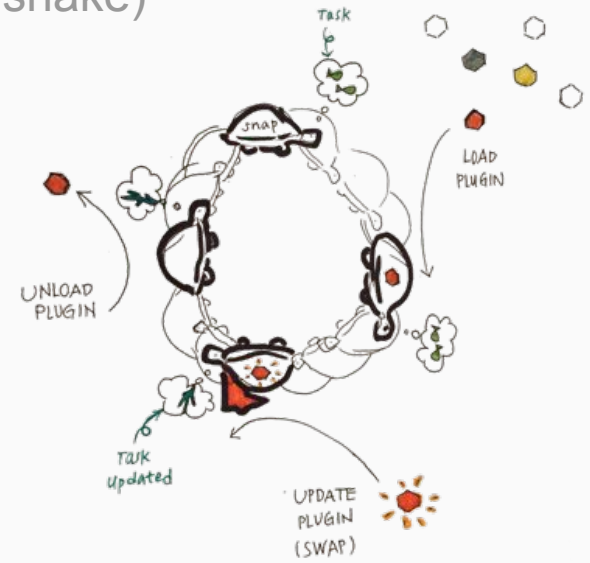
1. Snap starts the plugin
2. Snap negotiates with the plugin over stdout (handshake)



Plugin Authoring - Lifecycle

Loading a plugin

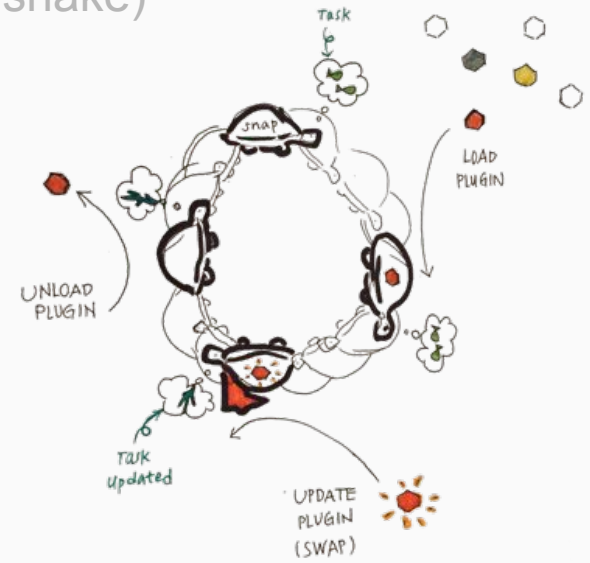
1. Snap starts the plugin
2. Snap negotiates with the plugin over stdout (handshake)
3. Snap calls the plugin
    a. To get its ConfigPolicy



Plugin Authoring - Lifecycle

Loading a plugin

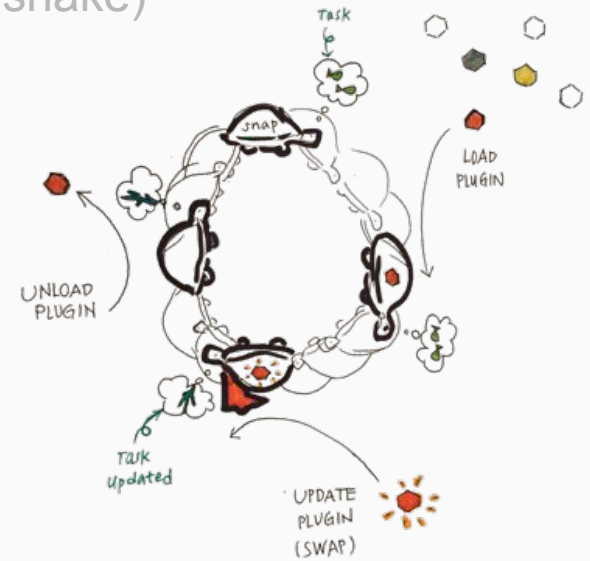
1. Snap starts the plugin
2. Snap negotiates with the plugin over stdout (handshake)
3. Snap calls the plugin
    a. To get its config policy
    b. To get what metrics it collects



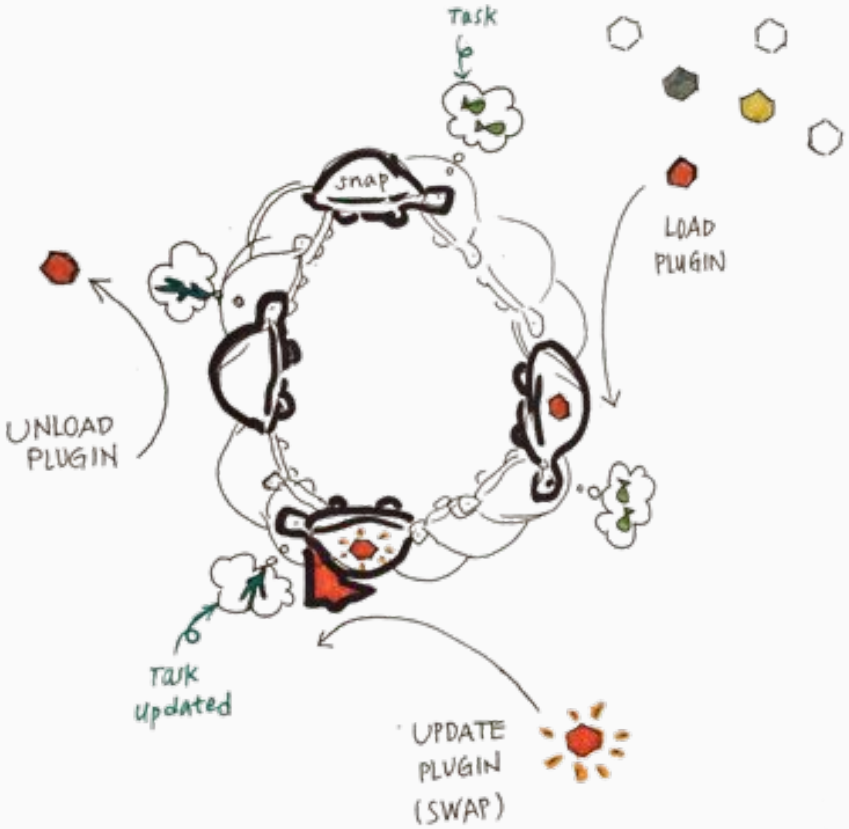
Plugin Authoring - Lifecycle

Loading a plugin

1. Snap starts the plugin
2. Snap negotiates with the plugin over stdout (handshake)
3. Snap calls the plugin
    a. To get its config policy
    b. To get what metrics it exposes
4. Snap stops the plugin



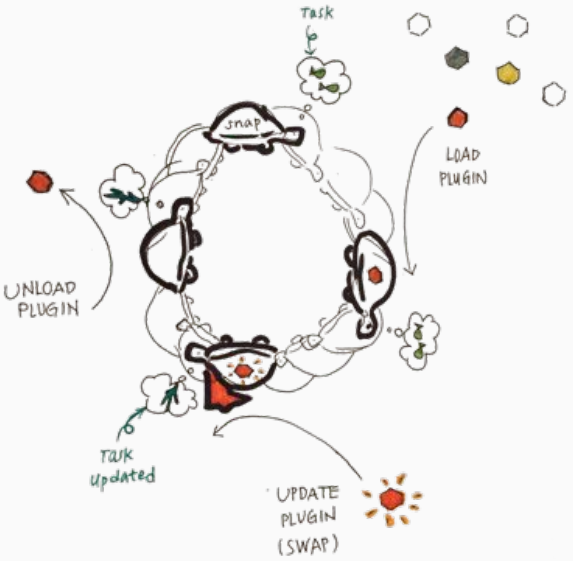
Starting a task



Plugin Authoring - Lifecycle

Starting a task

1. Snap ensures plugins required by the task are running



Plugin Authoring - Lifecycle

Starting a task

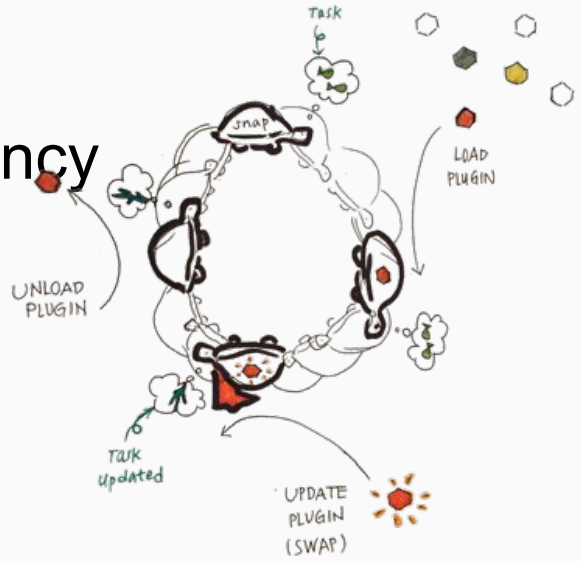
1. Snap ensures plugins required by the task are running
    a. Plugins that are used have their subscriptions increased

Plugin Authoring - Lifecycle

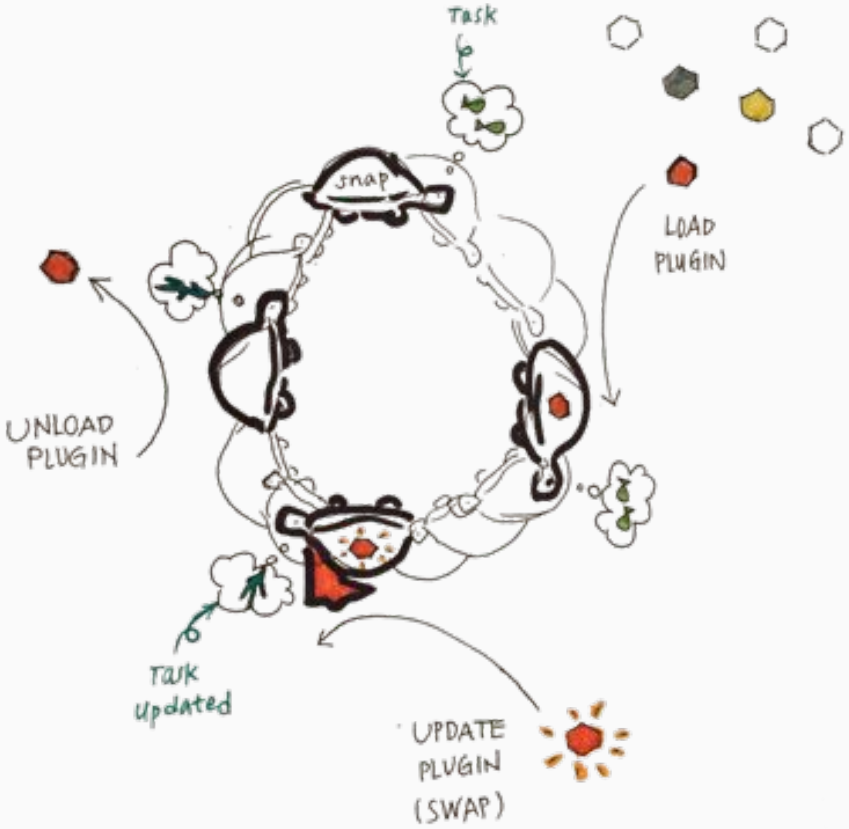
Starting a task

1. Snap ensures needed plugins required by the task are running
    a. Plugins that are used have their subscriptions increased

Plugin subscriptions drive the logic related to plugin routing and concurrency



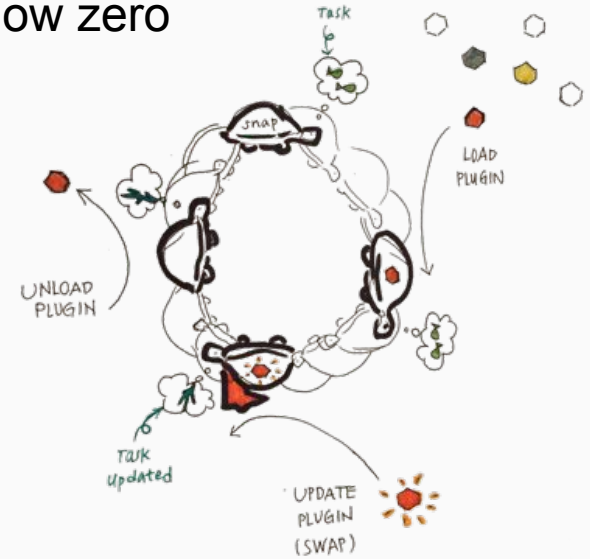
Stopping a task



Stopping a task

Causes plugins to stop

When the plugin's subscription count falls below zero



Plugin Meta

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

{"Meta":{**"Type":2,"Name":"graphite","Version":5,"RPCType":2,"RPCVersion":1,"ConcurrencyCount":5,"Exclusive":false,"CacheTTL":0,"RoutingStrategy":0**},"ListenAddress":"127.0.0.1:37166","PprofAddress":"0"}

Plugin Authoring - Meta

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - Plugin name as it will appear in the plugin catalog

Plugin Authoring - Meta

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher

Plugin Authoring - Meta

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin

Plugin Authoring - Meta

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used

Plugin Authoring - Meta

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
  - The default is ‘gRPC’
  - We will maintain legacy support for GORPC based plugins

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
- RPCVersion - Defines the version of the service the plugin implements

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
- RPCVersion - Defines the version of the service the plugin implements

```
service Collector {
    rpc CollectMetrics(MetricsArg) returns (MetricsReply) {}
    rpc GetMetricTypes(GetMetricTypesArg) returns (MetricsReply) {}
    rpc Ping(Empty) returns (ErrReply) {}
    rpc Kill(KillArg) returns (ErrReply) {}
    rpc GetConfigPolicy(Empty) returns (GetConfigPolicyReply) {}
}
```

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
- RPCVersion - Defines the version of the service the plugin implements

```
service Processor {
    rpc Process(PubProcArg) returns (MetricsReply) {}
    rpc Ping(Empty) returns (ErrReply) {}
    rpc Kill(KillArg) returns (ErrReply) {}
    rpc GetConfigPolicy(Empty) returns (GetConfigPolicyReply) {}
}
```

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
- RPCVersion - Defines the version of the service the plugin implements

```
service Publisher {
    rpc Publish(PubProcArg) returns (ErrReply) {}
    rpc Ping(Empty) returns (ErrReply) {}
    rpc Kill(KillArg) returns (ErrReply) {}
    rpc GetConfigPolicy(Empty) returns (GetConfigPolicyReply) {}
}
```

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
- RPCVersion - Defines the version of the service the plugin implements
- RoutingStrategy - Least Recently Used
  Sticky (task)
  Config based

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
- RPCVersion - Defines the version of the service the plugin implements
- RoutingStrategy - Least recently used, Sticky (task) or Config based
- ConcurrencyCount - Defines the maximum number of subscriptions the plugin can have before starting another instance of the plugin

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
- RPCVersion - Defines the version of the service the plugin implements
- RoutingStrategy - Least recently used, Sticky (task) or Config based
- ConcurrencyCount - Defines the maximum number of subscriptions the plugin can have before starting another instance of the plugin
- Exclusive - Results in a single instance of the plugin running for any number of subscriptions (ConcurrencyCount is ignored)

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
- RPCVersion - Defines the version of the service the plugin implements
- RoutingStrategy - Least recently used, Sticky (task) or Config based
- ConcurrencyCount - Defines the maximum number of subscriptions the plugin can have before starting another instance of the plugin
- Exclusive - Results in a single instance of the plugin running for any number of subscriptions
- CacheTTL - Overrides the default CacheTTL of 500ms

Plugin Meta

Is communicated over STDOUT between the plugin and Snap when a plugin starts

- Name - name as it will appear in the plugin catalog
- Type - collector, process, publisher
- Version - version of the plugin
- RPCType - RPC mechanism to be used
- RPCVersion - Defines the version of the service the plugin implements
- RoutingStrategy - Least recently used, Sticky (task) or Config based
- ConcurrencyCount - Defines the maximum number of subscriptions the plugin can have before starting another instance of the plugin
- Exclusive - Results in a single instance of the plugin running for any number of subscriptions
- CacheTTL - Overrides the default CacheTTL of 500ms
  If the framework attempts to call a plugin for a metric before it's TTL has expired it will be retrieved from the cache instead

Want to write a plugin?

Want to write a plugin?

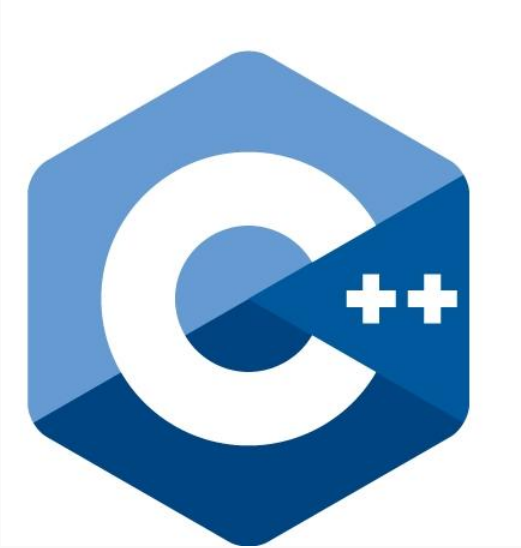
Start with one of the Snap plugin libs

Want to write a plugin?

Start with one of the Snap plugin libs

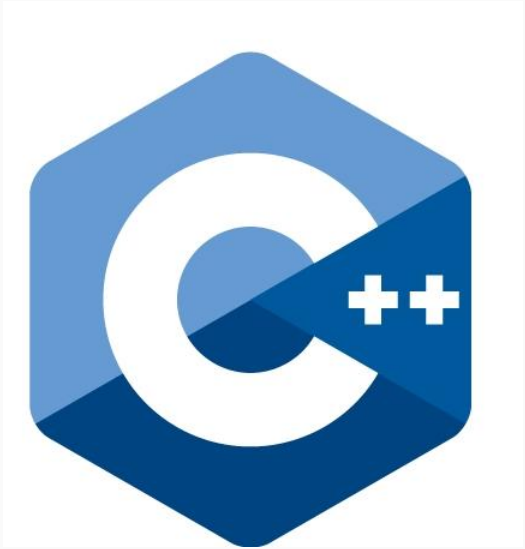
Want to write a plugin?

Start with one of the Snap plugin libs



Want to write a plugin?

Start with one of the Snap plugin libs

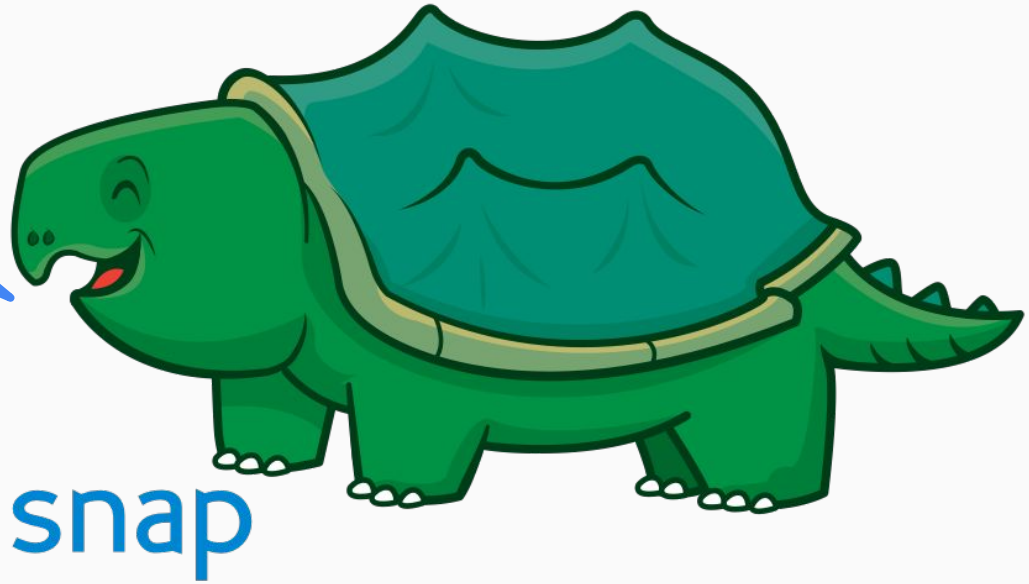


Plugin Authoring

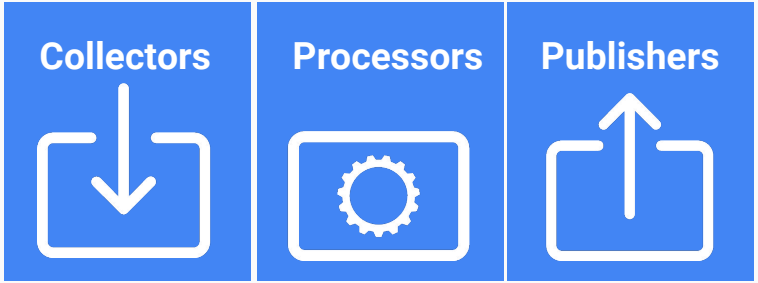
Leveraging existing investments

- Using the plugin-libs wrap existing tools
  - Example: snap-plugin-collector-diamond

To Review



The Snap Telemetry Framework



Snap is an Open Source telemetry framework that allows you to collect, process and publish measurements.

The Snap Telemetry Framework

Snap 1.0 is a general availability of a rock solid daemon with a collection of 67 plugins from multiple companies.

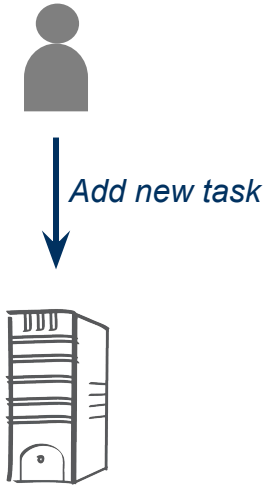
snap-telemetry.io

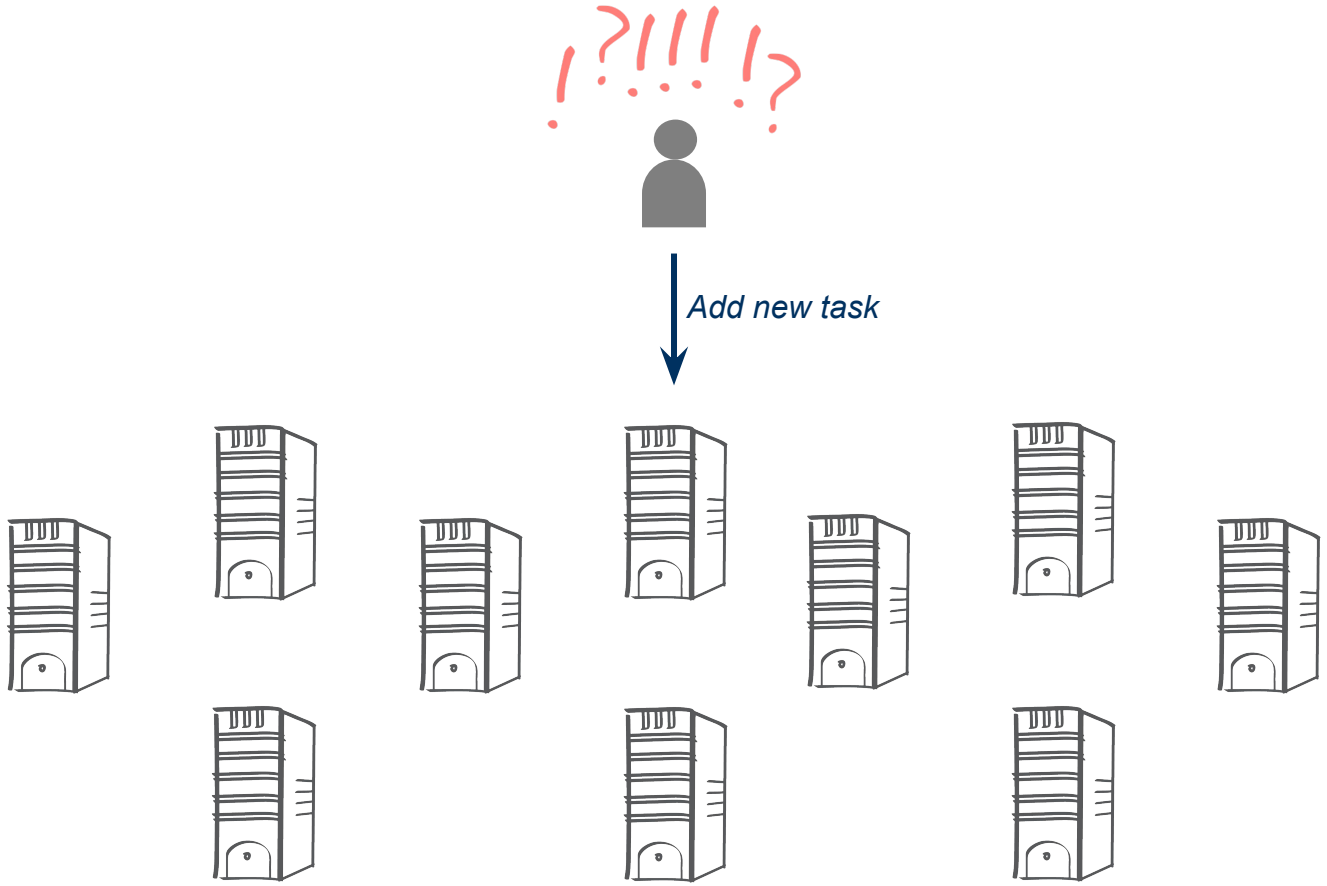
Snap 1.0 is the beginning. And we want you involved.

Thank you!

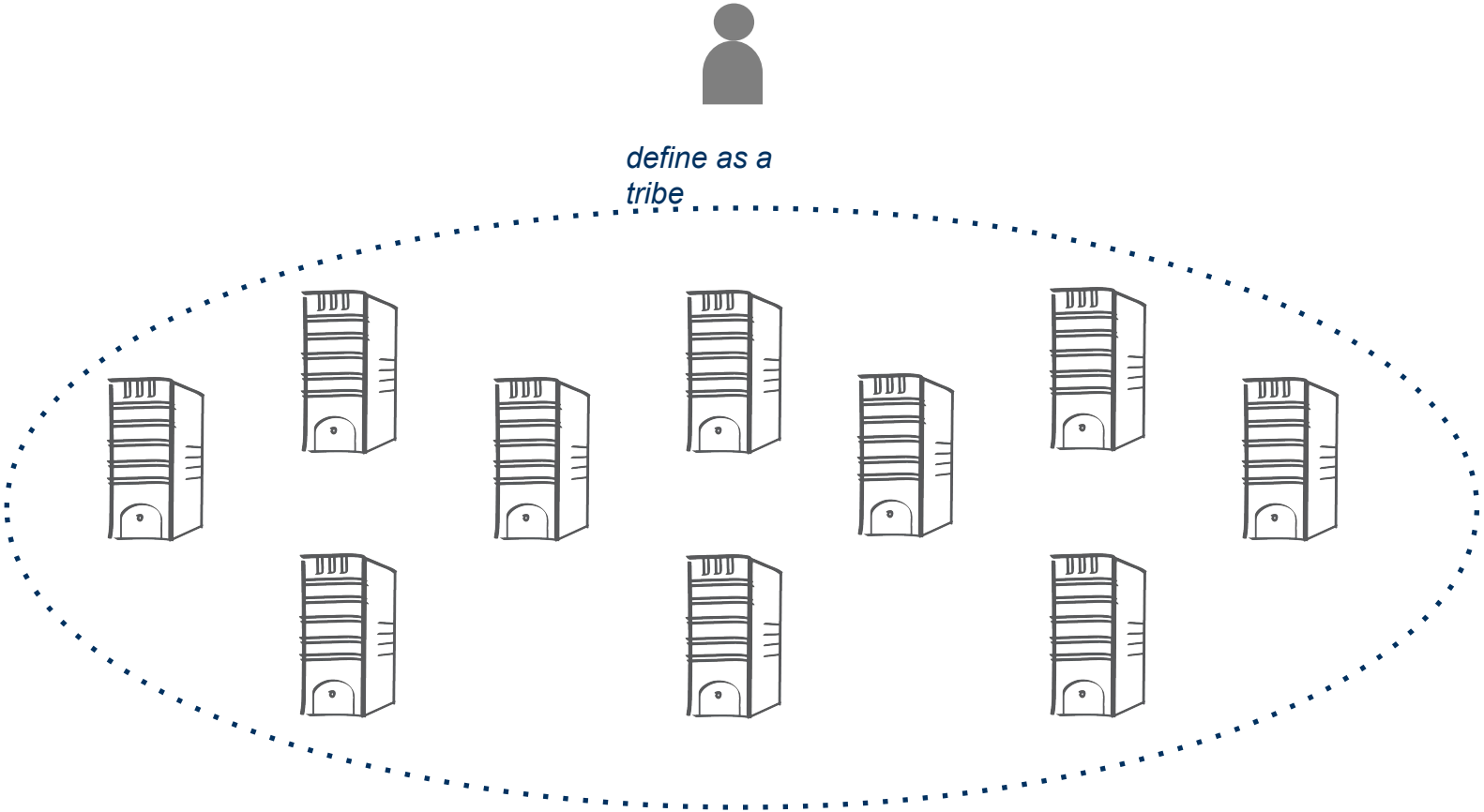
Backup Notes

Everything is Challenging At Scale

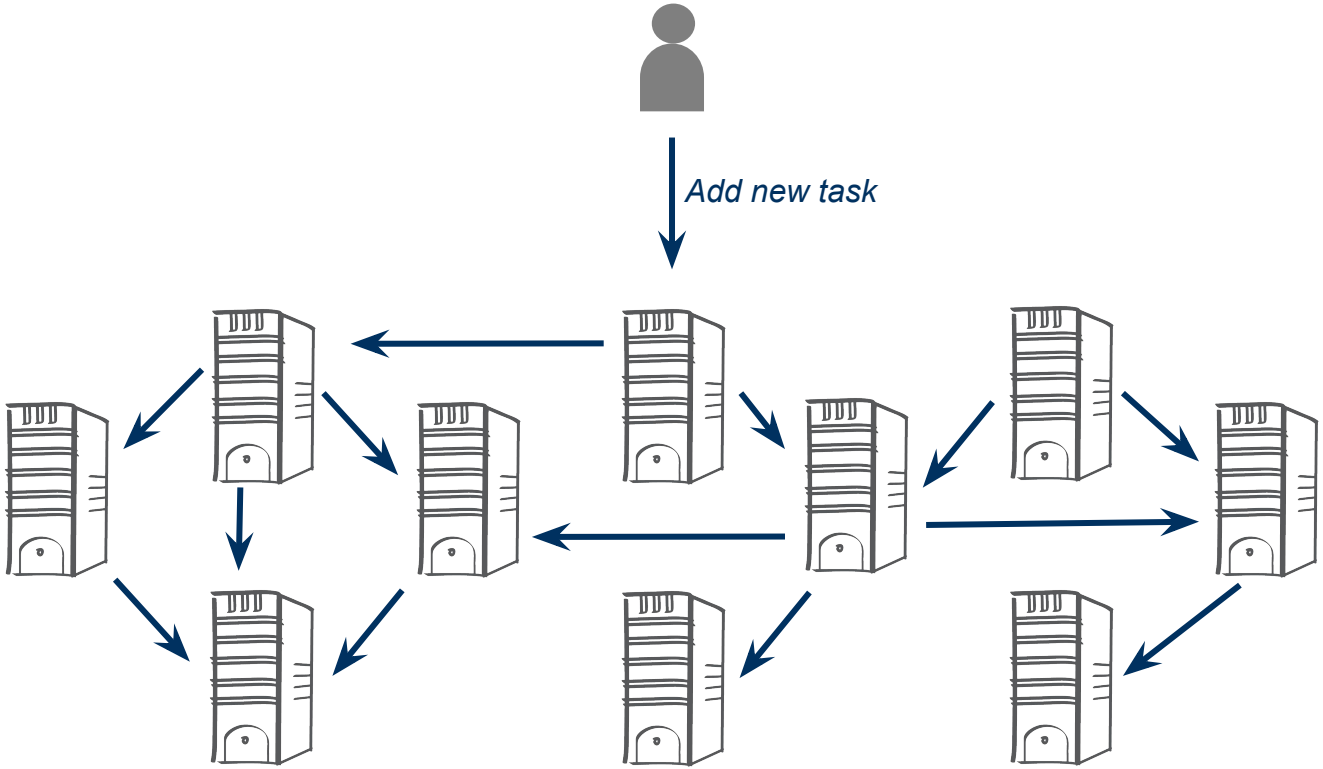

Add new task


!?!!!!?
Add new task

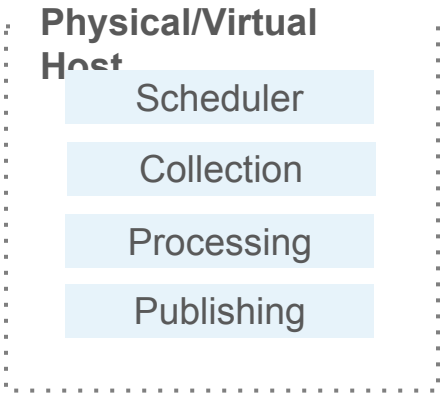
Scaling with Tribe



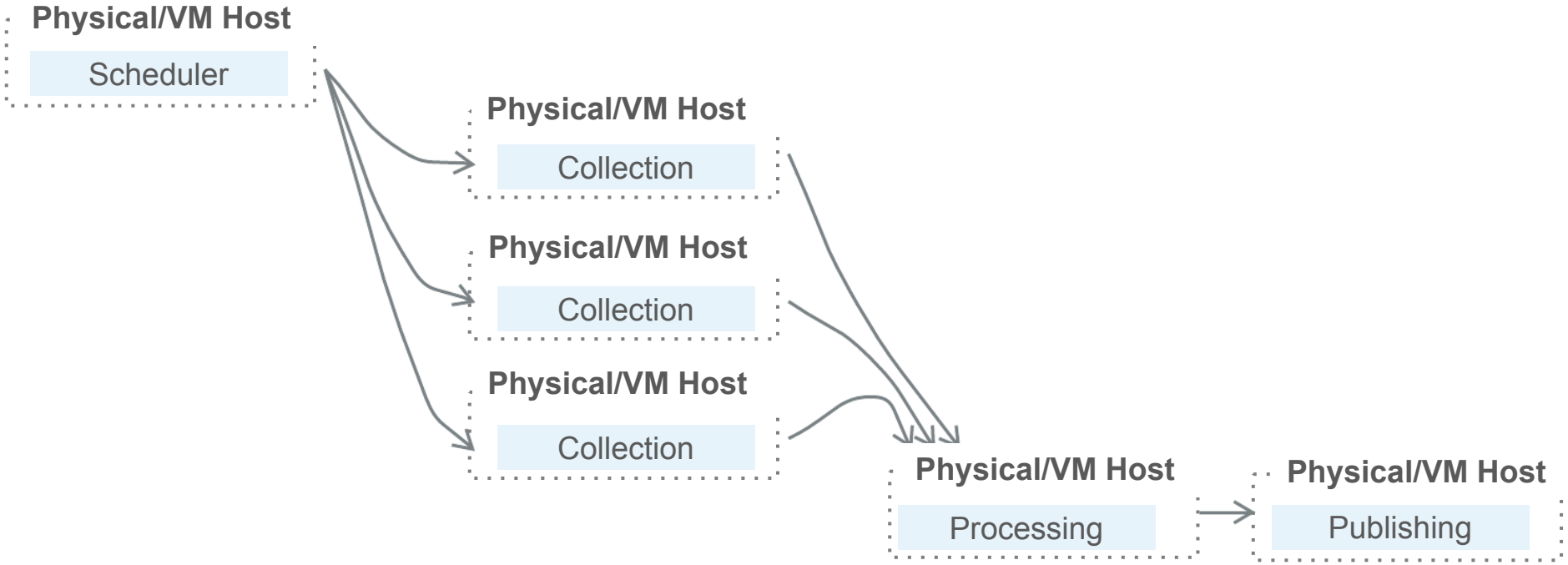
Scaling with Tribe



Snap | Distributed Workflow



Snap | Distributed Workflow



Snap | Overview – Example Workflows

Customizable definition of task and related workflow: