Building a Snap Telemetry Plugin and Visualizing the Data in Grafana

Jacob Lisi

Snap Telemetry Plugins

snapteld	
Collectors	• Collects Metrics
Processors	• Extends, and filters data
Publishers	• Publishes data to a target

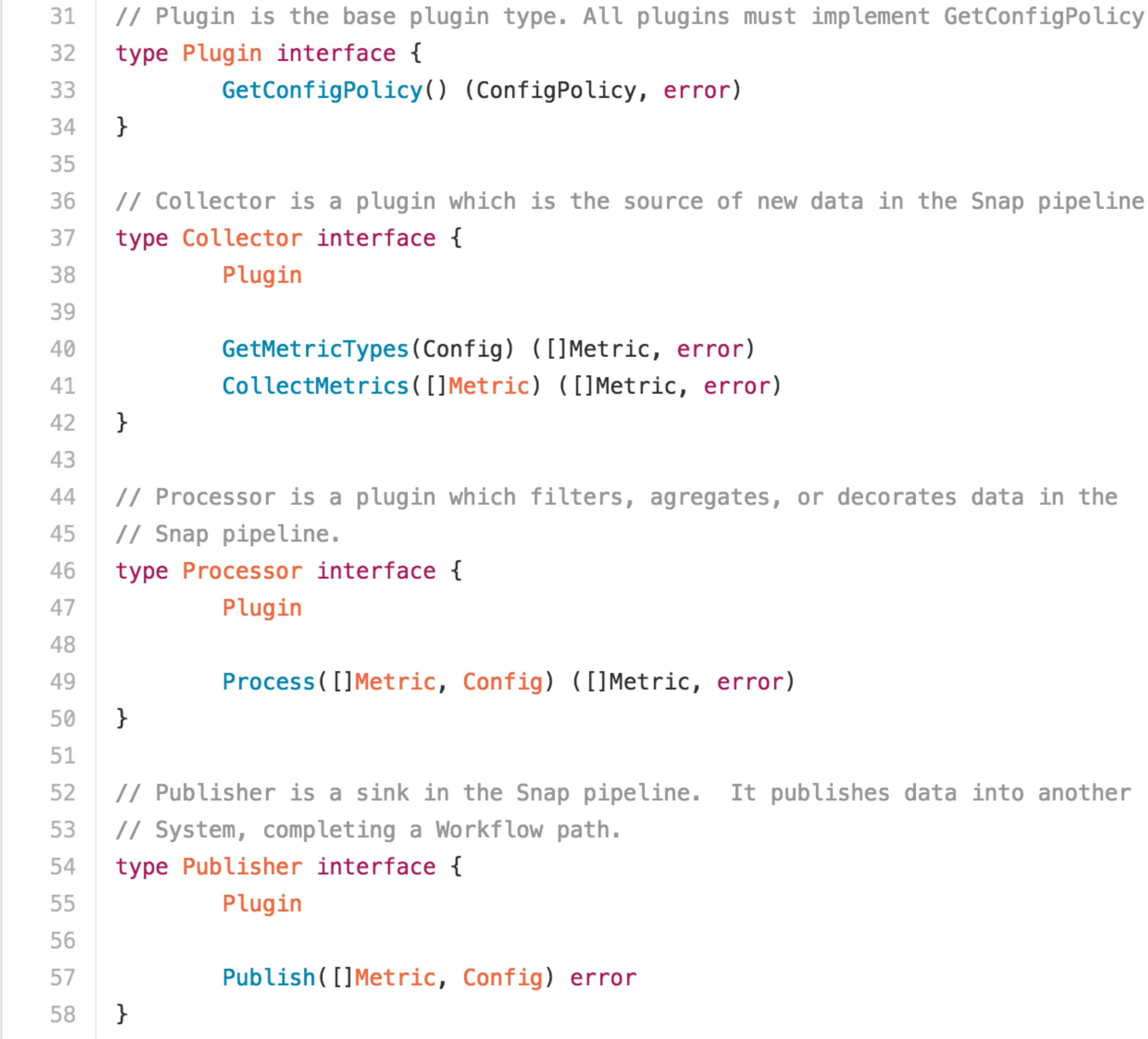
Writing a plugin

The best place to start:

Snap Plugin Authoring Guide

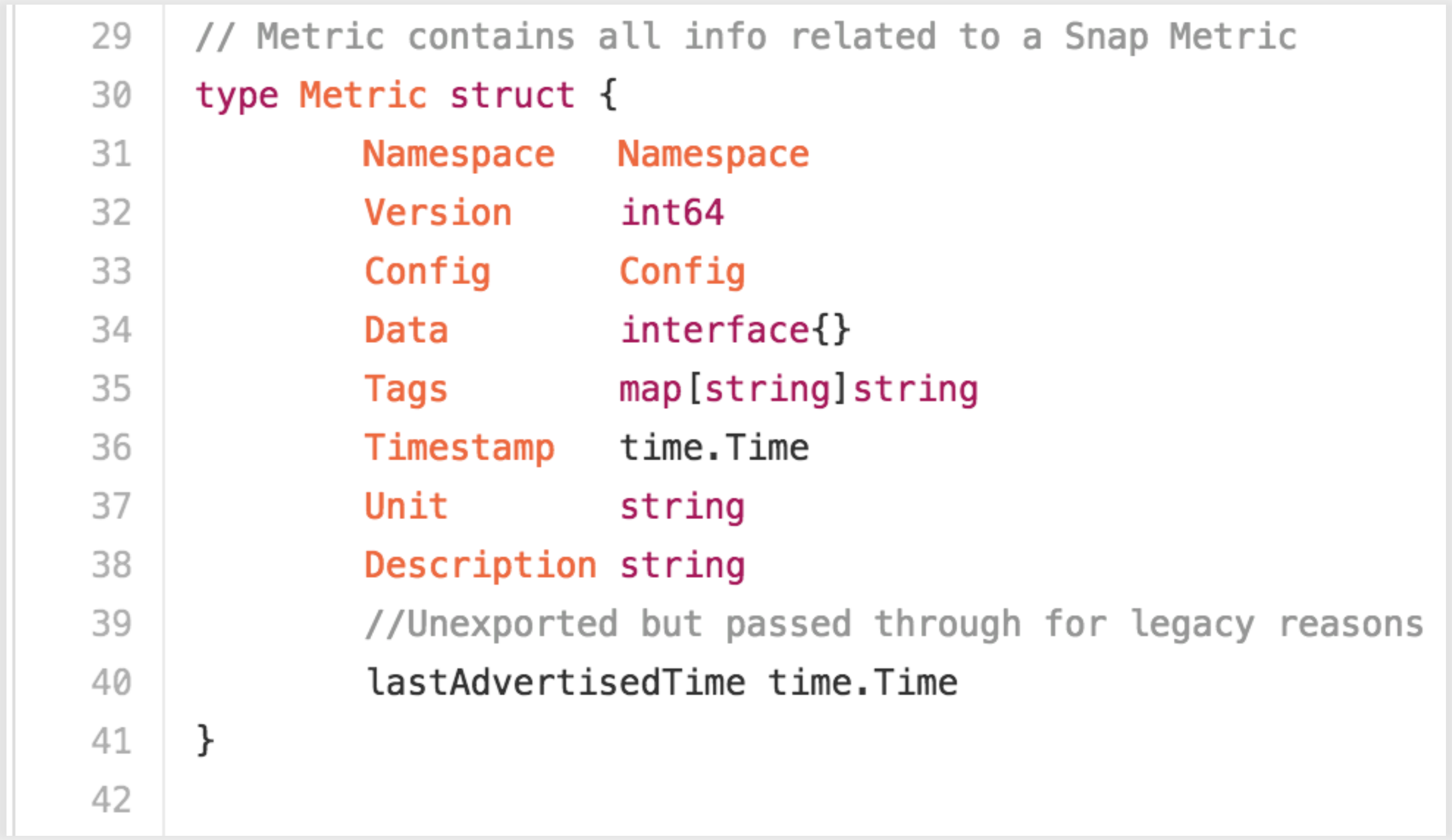
- Boilerplate
- Define your configuration
- Collector
  - Define your metrics
  - Collect the metrics
- Processor ~
- Publisher ~
- Test your plugin

Plugin Interfaces

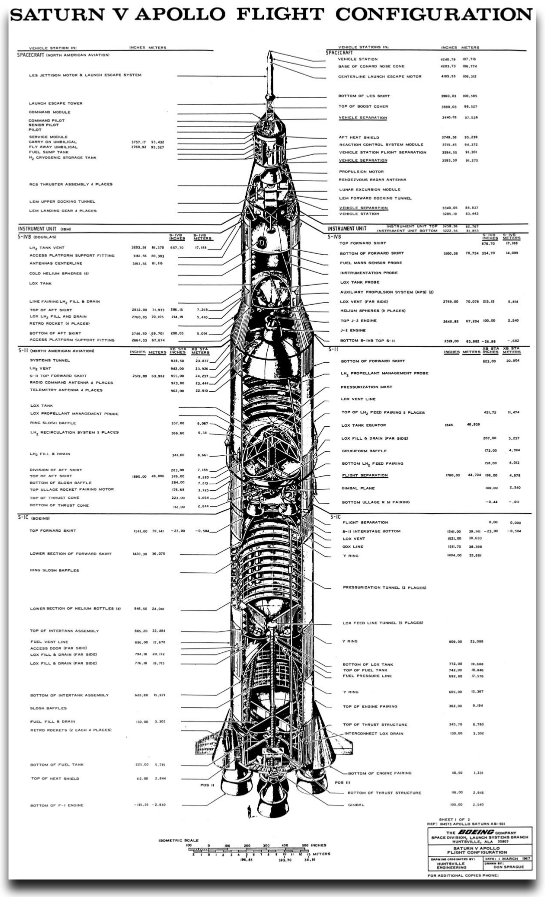


```go
// Plugin is the base plugin type. All plugins must implement GetConfigPolicy
type Plugin interface {
        GetConfigPolicy() (ConfigPolicy, error)
}

// Collector is a plugin which is the source of new data in the Snap pipeline
type Collector interface {
        Plugin

        GetMetricTypes(Config) ([]Metric, error)
        CollectMetrics([]Metric) ([]Metric, error)
}

// Processor is a plugin which filters, agregates, or decorates data in the
// Snap pipeline.
type Processor interface {
        Plugin

        Process([]Metric, Config) ([]Metric, error)
}

// Publisher is a sink in the Snap pipeline.  It publishes data into another
// System, completing a Workflow path.
type Publisher interface {
        Plugin

        Publish([]Metric, Config) error
}
```

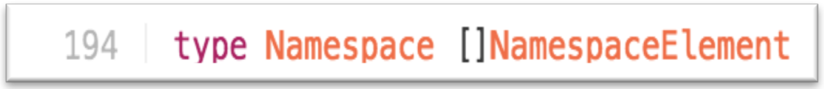
Data Model



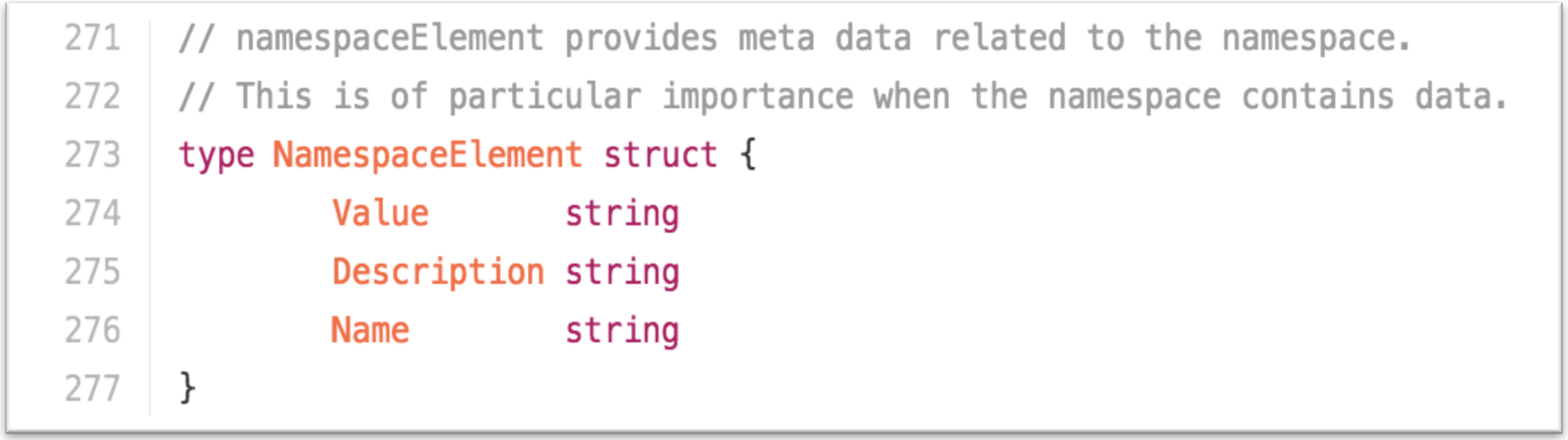
```go
// Metric contains all info related to a Snap Metric
type Metric struct {
        Namespace   Namespace
        Version     int64
        Config      Config
        Data        interface{}
        Tags        map[string]string
        Timestamp   time.Time
        Unit        string
        Description string
        //Unexported but passed through for legacy reasons
        lastAdvertisedTime time.Time
}
```



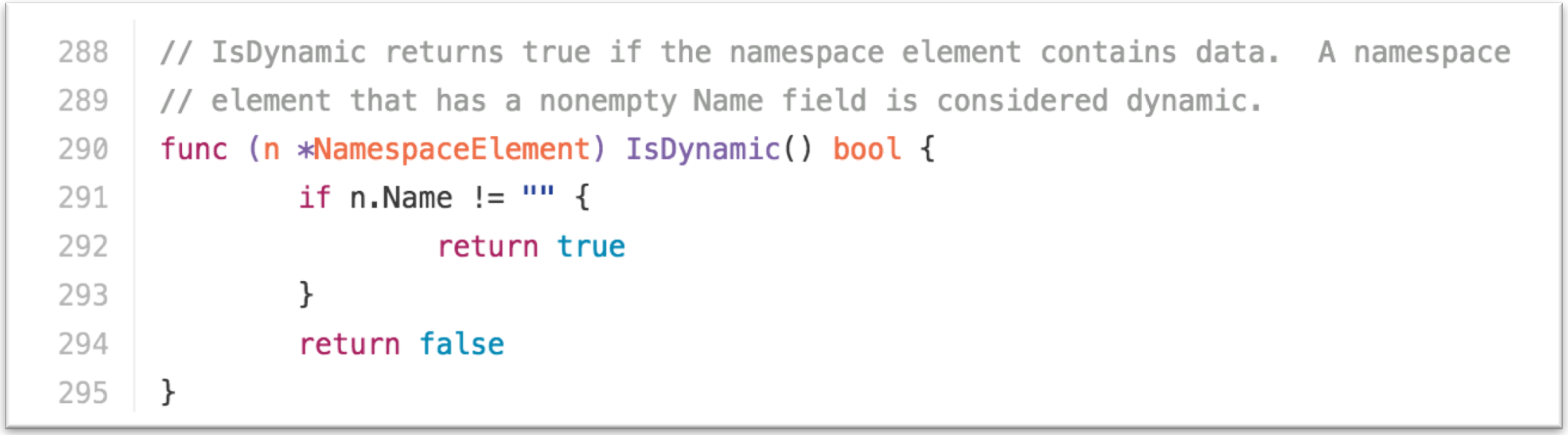
Metric Namespaces



```go
type Namespace []NamespaceElement
```



```go
// namespaceElement provides meta data related to the namespace.
// This is of particular importance when the namespace contains data.
type NamespaceElement struct {
        Value       string
        Description string
        Name        string
}
```



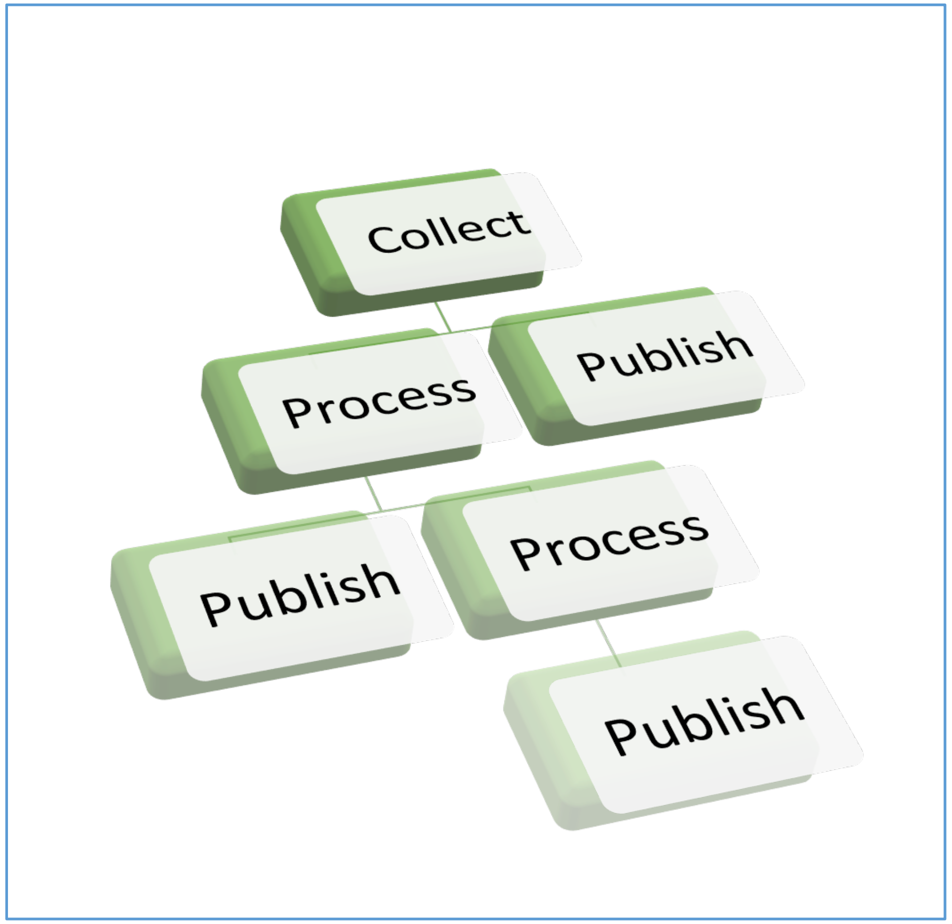
```go
// IsDynamic returns true if the namespace element contains data.  A namespace
// element that has a nonempty Name field is considered dynamic.
func (n *NamespaceElement) IsDynamic() bool {
        if n.Name != "" {
                return true
        }
        return false
}
```

Live Demo

- Writing a simple collector for a json endpoint
- Visualize our metrics in grafana
- NOTICE
  - Please forgive me for any errors that may occur

Snap Task Workflows

- Can be written in json or yaml
- Two Parts:
  - Header
    - Version
    - Schedule
    - Max-Failures
    - Deadline
  - Workflow
    - Directed acyclic graph
    - Begins with a collect and is followed by and number of process and publish directives
    - Process and Publish jobs can be forwarded to remote snap nodes



Testing Your Plugin

Refactoring without Tests

Sure I can write some test cases

The Gold Standard

Lessons Learned

1. Watching a task only gives visibility into the collection layer
2. Be sure to pass your tags/configs from the task specified metrics to the collected metrics
3. Profile your plugin before deploying widely!
    1. Too much io/network_usage/etc
4. Dynamic namespacing makes everyone’s life better

Questions?