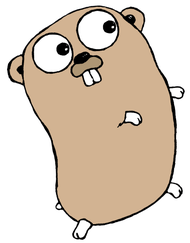
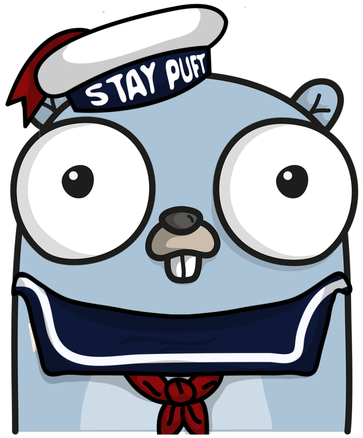
The Kubernetes Go Client Package



About me

YUNAR

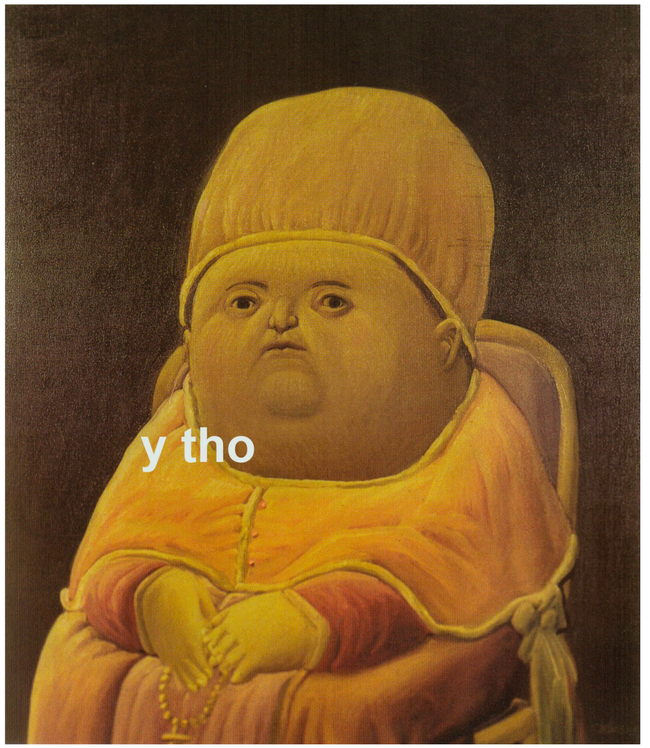


Why this talk?

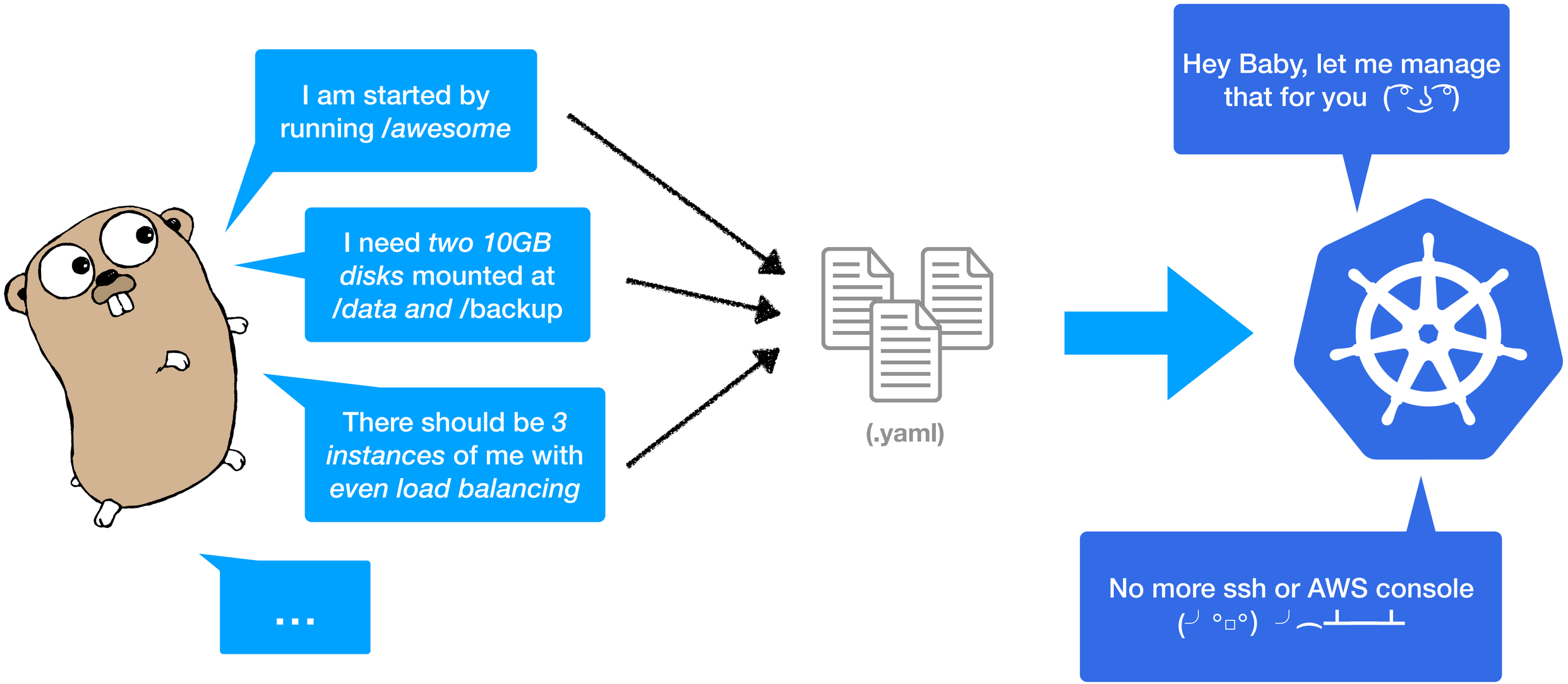
1. It’s actually quite easy to automate k8s infrastructure tasks using the client package

⇒ We are now switching from sophisticated/ complicated CI pipelines to k8s operators

2. It’s super easy to adapt your go application to take advantage of some neat k8s features



K8S explained in 60sec




S GET ST
makeameme.org

Installation Note

Versioning matters!

	Kubernetes 1.7	Kubernetes 1.8	Kubernetes 1.9	Kubernetes 1.10	Kubernetes 1.11	Kubernetes 1.12	Kubernetes 1.13
client-go 4.0	✓	+-	+-	+-	+-	+-	+-
client-go 5.0	+-	✓	+-	+-	+-	+-	+-
client-go 6.0	+-	+-	✓	+-	+-	+-	+-
client-go 7.0	+-	+-	+-	✓	+-	+-	+-
client-go 8.0	+-	+-	+-	+-	✓	+-	+-
client-go 9.0	+-	+-	+-	+-	+-	✓	+-
client-go 10.0	+-	+-	+-	+-	+-	+-	✓
client-go HEAD	+-	+-	+-	+-	+-	+-	+-

You can find the k8s version of your cluster by running *kubectl version*

Installation Note

Versioning matters! Two cool ways to do it…

Go Dependency Manager

1. Add the following to your *Gopkg.toml*

```
[[constraint]]
  name = "k8s.io/client-go"
  version = “v10.0.0"
```

2. Run *dep ensure*

Go Modules

1. Run *go mod init <great-name>*

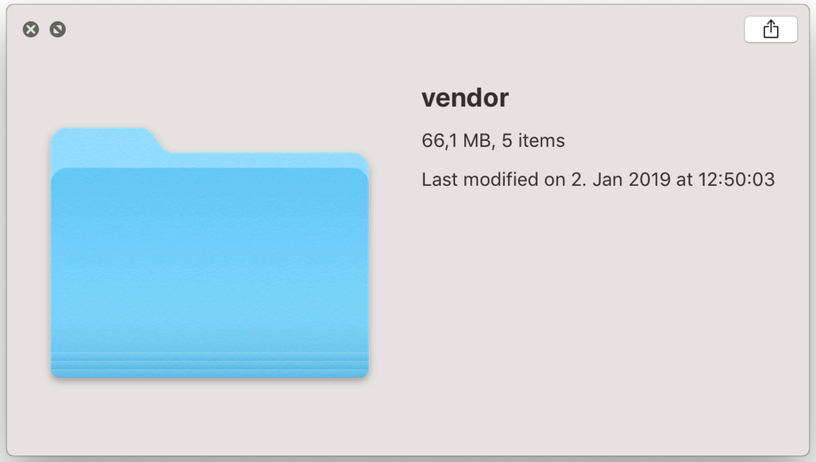
2. Add the following to your *go.mod*

```
require (
  k8s.io/client-go v10.0.0
  k8s.io/api kubernetes-1.13.2
)
```

3. Run *go build*

Installation Note

28 packages later…



Don't Try This at Home

You can run your own kubernetes cluster on your computer using *minikube*

https://kubernetes.io/docs/tasks/tools/install-minikube/


LET'S GET STARTED
makeameme.org

Connecting to the k8s api server

1. From your local machine
2. From within the k8s cluster

Connecting to the k8s api server

Locally

```go
package main

import (
  "k8s.io/client-go/rest"
  "k8s.io/client-go/tools/clientcmd"
  "k8s.io/client-go/kubernetes"
  "path/filepath"
  "os"
)

func main() {
  var config *rest.Config

  kubecfgpath := filepath.Join(os.Getenv("HOME"), ".kube", "config")
  config, err := clientcmd.BuildConfigFromFlags("", kubecfgpath)

  clientset, err := kubernetes.NewForConfig(config)

  ...
}
```

Connecting to the k8s api server

Locally - What happens behind the scenes

~/.kube/config


```yaml
apiVersion: v1
kind: Config
clusters:
- name: minikube
  cluster:
    insecure-skip-tls-verify: true
    server: https://192.168.43.66:6443
contexts:
- context:
    cluster: local
    user: admin
  name: minikube
current-context: minikube
users:
- name: admin
  user:
    password: admin
    username: secretpassword
```

Connecting to the k8s api server

Locally - Tips and Tricks 💡

1. Make sure your kubeconfig points to the correct k8s context!

2. Make sure your user has the correct access rights

```yaml
apiVersion: v1
kind: Config
clusters:
- name: minikube
  cluster:
    insecure-skip-tls-verify: true
    server: https://192.168.43.66:6443
contexts:
- context:
    cluster: local
    user: admin
  name: minikube
current-context: minikube
users:
- name: admin
  user:
    password: admin
    username: secretpassword
```

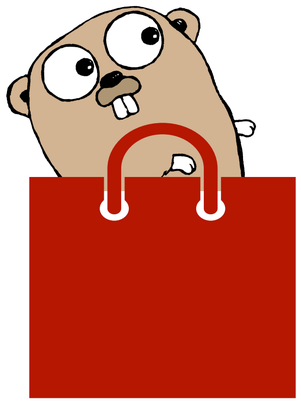
Connecting to the k8s api server

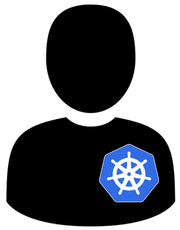
Within the k8s cluster

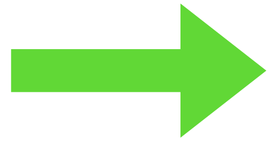
```go
import (
  "k8s.io/client-go/kubernetes"
  "k8s.io/client-go/rest"
)


func main() {
  var config *rest.Config

  config, err := rest.InClusterConfig()

  if err != nil {
    panic(err)
  }

  clientset, err := kubernetes.NewForConfig(config)

  ...
}
```

Connecting to the k8s api server

Within the k8s cluster - What happens behind the scenes





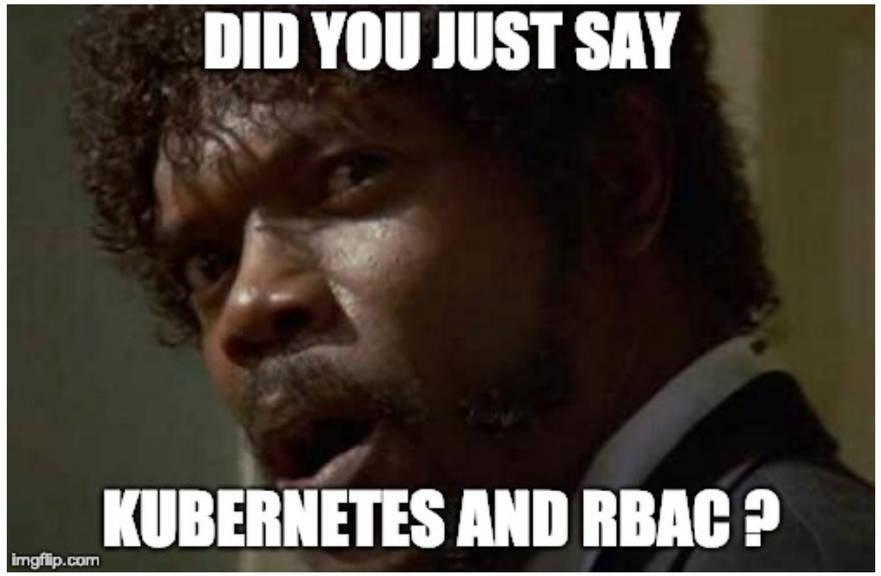


k8s service account

```
...

spec:
  containers:
    - ...
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-knx5r
      readOnly: true
...
```

Pod

Connecting to the k8s api server



Connecting to the k8s api server

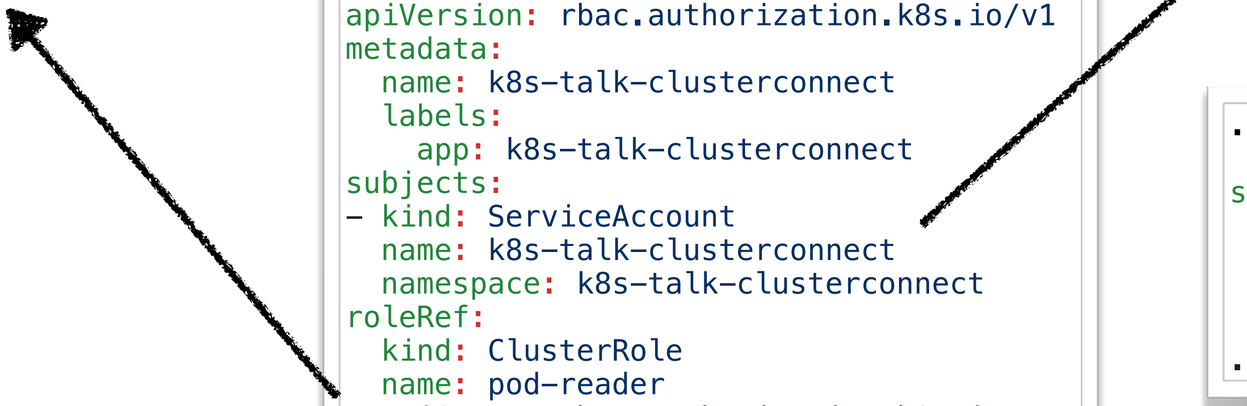
Within the k8s cluster - The holy trinity of k8s RBAC

```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: pod-reader
  labels:
    app: k8s-talk-clusterconnect
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

[Cluster]Role

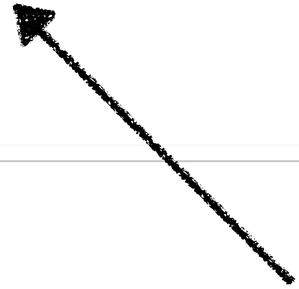
```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name:  k8s-talk-clusterconnect
  labels:
    app: k8s-talk-clusterconnect
```

ServiceAccount



```yaml
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: k8s-talk-clusterconnect
  labels:
    app: k8s-talk-clusterconnect
subjects:
- kind: ServiceAccount
  name: k8s-talk-clusterconnect
  namespace: k8s-talk-clusterconnect
roleRef:
  kind: ClusterRole
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

[Cluster]Rolebinding



```yaml
...

spec:
  template:
    spec:
      serviceAccountName: k8s-talk-clusterconnect

...
```

Deployment

Interacting with the k8s api

Fetching k8s API Objects

Example - Fetching all Pods

```go
package getallpods

import (
  "os"
  "path/filepath"

  "fmt"

  metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
  "k8s.io/client-go/kubernetes"
  "k8s.io/client-go/rest"
  "k8s.io/client-go/tools/clientcmd"
)

func main() {
  ...

  allPods, err := clientset.CoreV1().Pods("").List(metav1.ListOptions{})
  if err != nil {
    panic(err)
  }
  for _, pod := range allPods.Items {
    fmt.Println(pod.ObjectMeta.Name)
  }
}
```

Pod Templates

Pod templates are pod specifications which are included in other objects, such as Replication Controllers, Jobs, and DaemonSets. Controllers use Pod Templates to make actual pods. The sample below is a simple manifest for a Pod which contains a container that prints a message.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
```

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
```

Creating k8s objects

Example - Creating a secret

```
package main

import (
  ...
  corev1 "k8s.io/api/core/v1"
)

func main() {
    ...

    secret := &corev1.Secret{

      ObjectMeta: metav1.ObjectMeta {
        Name: "k8s-talk-createsecret",
        Namespace: "k8s-talk-createsecret",
        Labels: map[string]string {
          "app": "k8s-talk-createsecret",
        },
      },

      StringData: map[string]string {
        "username": "simon",
        "password": "secretpw123",
      },
    }

    res, err := clientset.CoreV1().Secrets(secret.ObjectMeta.Namespace).Create(secret)
   ...
}
```

One more thing…

Watch - The k8s way of fetching objects

```
package main

import (
  “k8s.io/apimachinery/pkg/watch"
  ...
)

func main() {
  ...
  cmns, cmname := "simons-ns", "simons-configmap"
  listOptions := metav1.ListOptions{
    FieldSelector: fmt.Sprintf("metadata.name==%s", cmname),
  }

  log.Printf("Establishing watch on configmap '%s/%s'", cmns, cmname)
  watcher, err := clientset.CoreV1().ConfigMaps(cmns).Watch(listOptions)

  if err != nil {
    fmt.Printf("Could not establish watch on '%s/%s': %v", cmns, cmname, err)
  }

  for event := range watcher.ResultChan() {
    switch event.Type {
    case watch.Added:
      log.Printf("configmap '%s/%s' was created", cmns, cmname)
    case watch.Modified:
      log.Printf("configmap '%s/%s' was modified", cmns, cmname)
    case watch.Deleted:
      log.Printf("configmap '%s/%s' was deleted :(", cmns, cmname)
    }
  }
}
```

Some cool k8s-go-client projects to explore

Tumblr Secret Projector
https://github.com/tumblr/k8s-secret-projector

Istio Sidecar Injector
https://github.com/istio/istio/blob/master/pilot/pkg/kube/inject/webhook.go

Image Credits

Beard made by Freepik from www.flaticon.com licensed by CC 3.0 BY

Document made by Yanni Lung from www.iconfinder.com licensed by Free for commercial use

Man User free icon made by Freepik from www.flaticon.com is licensed by CC 3.0 BY

Follow me

SimonTheLeg

@SimonTheLeg