Open Cloud

Sobre mim

- 7 anos na área de TI
- 5,5 anos como desenvolvedor
- 1,5 anos como tech leader
- Atualmente trabalho como tech leader na Nuveo

@_felipeweb
https://felipeweb.dev

Estamos contratando!!!

Enviar email para talentos@nuveo.ai

O que é Open Cloud

É uma maneira de unificar as clouds fazendo com que nossas aplicações não sintam de diferença se estão usando serviços do provider A ou provider B

https://open-cloud-foundation.org/

Porque Open Cloud

- Sem vendor lock-in
- Resiência
- Código portável

Como adotar o Open Cloud?

Especificação

http://occi-wg.org/about/specification/


Woo-Hoo!

E os grandes providers do mercado?


JUDGES

Recomeço

Premissas

- Um serviço por vez
- Começar pelo serviço mais usado
- Facil de usar
- Não perder especificidades dos providers

Escolhas

- AWS & GCP
- Serviço de storage

Código

Upload para o S3

nuveo

```go
// Upload an image to amazon S3
func uploadFile() {

  file, err := os.Open("./images/test.jpeg")
  if err != nil {
    log.Fatal("Failed to open file :", err)
  }
  uploader := s3manager.NewUploader(session.New(&aws.Config{Region: aws.String("us-east-1")}))
  result, err := uploader.Upload(&s3manager.UploadInput{
    Body : file,
    Bucket: aws.String(bucket),
    Key : aws.String(key),
  })
  if err != nil {
    log.Fatalln("Failed to upload ", err)
  }
  log.Println("Success Upload to ", result.Location)
}
```

Upload GCS

nuveo

```go
// Upload an image to amazon GCS
func uploadFile() {

  file, err := os.Open("./images/test.jpeg")
  if err != nil {
    log.Fatal("Failed to open file :", err)
  }
  wc := client.Bucket(bucket).Object(object).NewWriter(ctx)
  if _, err = io.Copy(wc, file); err != nil {
    log.Fatalln("Failed to upload ", err)
  }
  if err := wc.Close(); err != nil {
    log.Fatalln("Failed to upload ", err)
  }
  log.Println("Success Upload ")
}
```

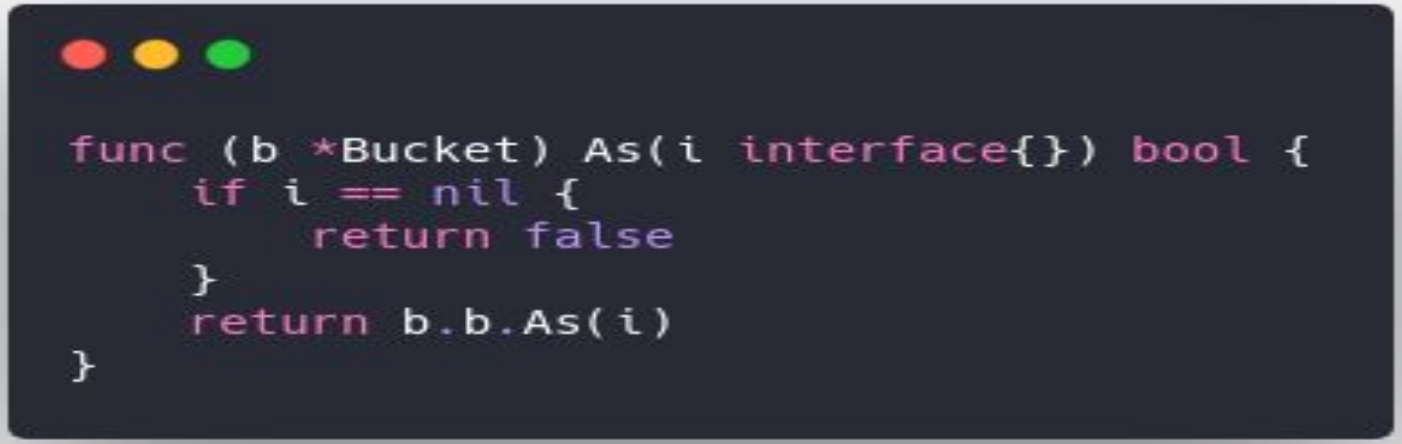
Como Padronizar

nuveo

```go
type Bucket struct {
    b       driver.Bucket
    mu      sync.RWMutex
    closed bool
}

func (b *Bucket) WriteAll(ctx context.Context, key string, p []byte) error {
  w, err := b.NewWriter(ctx, key)
    if err != nil {
        return err
    }
    if _, err := w.Write(p); err != nil {
        _ = w.Close()
        return err
    }
    return w.Close()
}
```

Não perder especificidades do provider



```go
func (b *Bucket) As(i interface{}) bool {
    if i == nil {
        return false
    }
    return b.b.As(i)
}
```

Não perder especificidades do provider

nuveo

```go
func Upload(ctx context.Context, bucket, fileName, region, acl string, byt []byte) (err error) {
    c := &aws.Config{
        Region:      aws.String(region),
        Credentials: credentials.NewEnvCredentials(),
    }
    s, err := session.NewSession(c)
    if err != nil {
        return
    }
    b, err := s3blob.OpenBucket(ctx, bucket, s, nil)
    if err != nil {
        return
    }
    before := func(asFunc func(interface{}) bool) error {
        req := &s3manager.UploadInput{}
        ok := asFunc(&req)
        if !ok {
            return errors.New("invalid s3 type")
        }
        req.ACL = aws.String(acl)
        return nil
    }
    w, err := b.NewWriter(ctx, appendTime(fileName), &blob.WriterOptions{
        ContentType: forceContentTypeByExtension(fileName),
        BeforeWrite: before,
    })
    if err != nil {
        return
    }
    defer w.Close()
    _, err = w.Write(byt)
    return
}
```

Autenticando no provider

nuveo

```go
func (mux *URLMux) OpenBucket(ctx context.Context, urlstr string) (*Bucket, error) {
    opener, u, err := mux.schemes.FromString("Bucket", urlstr)
    if err != nil {
        return nil, err
    }
    return applyPrefixParam(ctx, opener.(BucketURLOpener), u)
}
```

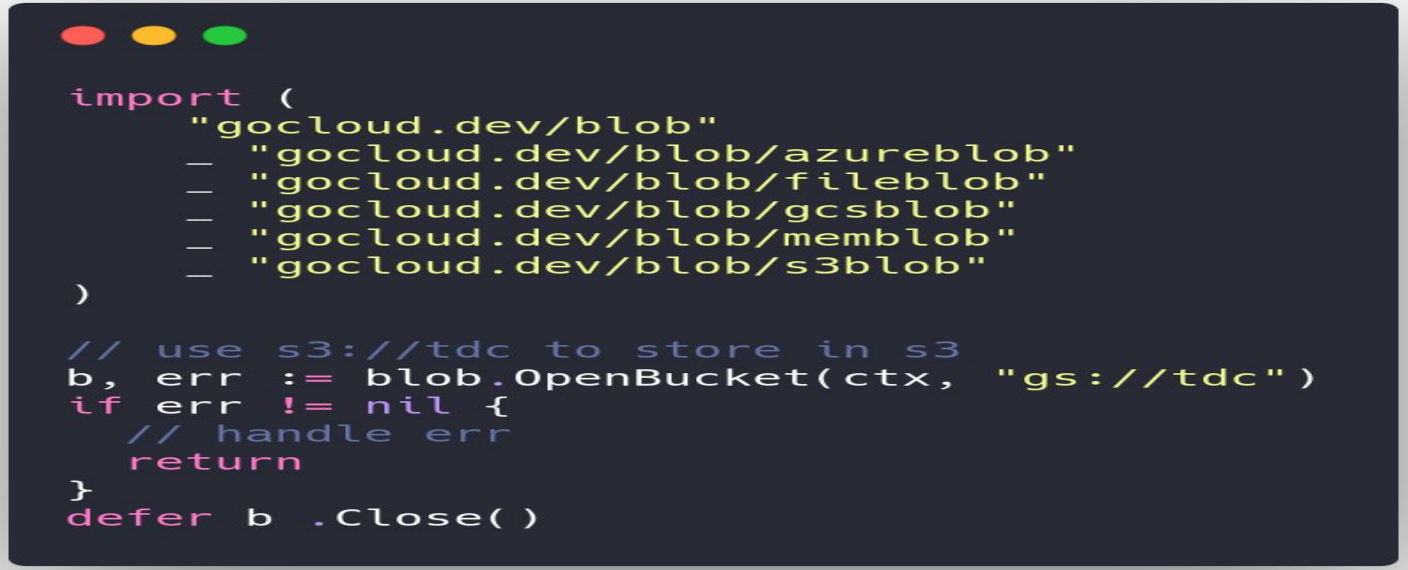
Autenticando no provider

nuveo

```go
func init() {
    blob.DefaultURLMux().RegisterBucket(Scheme, new(lazySessionOpener))
}

type lazySessionOpener struct {
    init   sync.Once
    opener *URLOpener
    err    error
}
func (o *lazySessionOpener) OpenBucketURL(ctx context.Context, u *url.URL) (*blob.Bucket, error) {
    o.init.Do(func() {
        sess, err := aws.NewDefaultSession()
        if err != nil {
            o.err = err
            return
        }
        o.opener = &URLOpener{
            ConfigProvider: sess,
        }
    })
    if o.err != nil {
        return nil, fmt.Errorf("open bucket %v: %v", u, o.err)
    }
    return o.opener.OpenBucketURL(ctx, u)
}
const Scheme = "s3"
```

Como usar



```
import (
      "gocloud.dev/blob"
      _ "gocloud.dev/blob/azureblob"
      _ "gocloud.dev/blob/fileblob"
      _ "gocloud.dev/blob/gcsblob"
      _ "gocloud.dev/blob/memblob"
      _ "gocloud.dev/blob/s3blob"
)

// use s3://tdc to store in s3
b, err := blob.OpenBucket(ctx, "gs://tdc")
if err != nil {
   // handle err
   return
}
defer b .Close( )
```

Dúvidas???

Obrigado!!!

@_felipeweb
https://felipeweb.dev

Estamos contratando!!!

Enviar email para talentos@nuveo.ai