Taylor Thomas
Senior Software Engineer
Microsoft Azure

The Rusty Boat

Krustlet and Kubernetes

Who am I?

- Helm Core Maintainer
- Doing Kubernetes since 1.2 and Docker since 0.7
- New Rustacean
- Social media handles
  - Twitter: @_oftaylor
  - GitHub: @thomastaylor312
  - Kubernetes Slack: @oftaylor

Deck Map

- Deck 4: The Crow’s Nest
- Deck 3: The Pool Deck
- Deck 2: Entertainment District
- Deck 1: The Engine Room

The Crow’s Nest

What's this WASM thing?

WASI

More acronyms!

- This stands for Web Assembly System Interface
- It even has a landing page! https://wasi.dev
- This is meant to be a standard for interacting with a host system, no matter the OS
- It is VERY new and not fully fleshed out

OCI, CRI, Oh My

More TLAs!

- OCI is the Open Container Initiative and has to do with all things containers
- CRI is the Container Runtime Interface, an API defined by Kubernetes that all container runtimes must implement
- Virtual Kubelet is something that masks as a normal Kubernetes Kubelet but exposes another provider

Why did we make this?

- Security
- Density
- More control
- Actually “run everywhere”
- Smaller footprint for embedded devices

Pool Deck

@_oftaylor

Krustlet

- Kubernetes RUST kubLET
- Its primary purpose is to run WASM modules within Kubernetes
- Multiple Providers

The Features

What's there

- Basic pod lifecycle
- Downward API support
- Environment variables (including Secrets/ConfigMaps)
- HostPath, Secret, and ConfigMap volumes

What's not there

- ARM
- Windows (kinda)
- Init Containers
- Cloud provider volume types
- Eventing and all the conditions

The Providers

waSCC

- Actor Model with capability hot swapping
- Has network support
- Strong security model on top of normal WASM modules

WASI

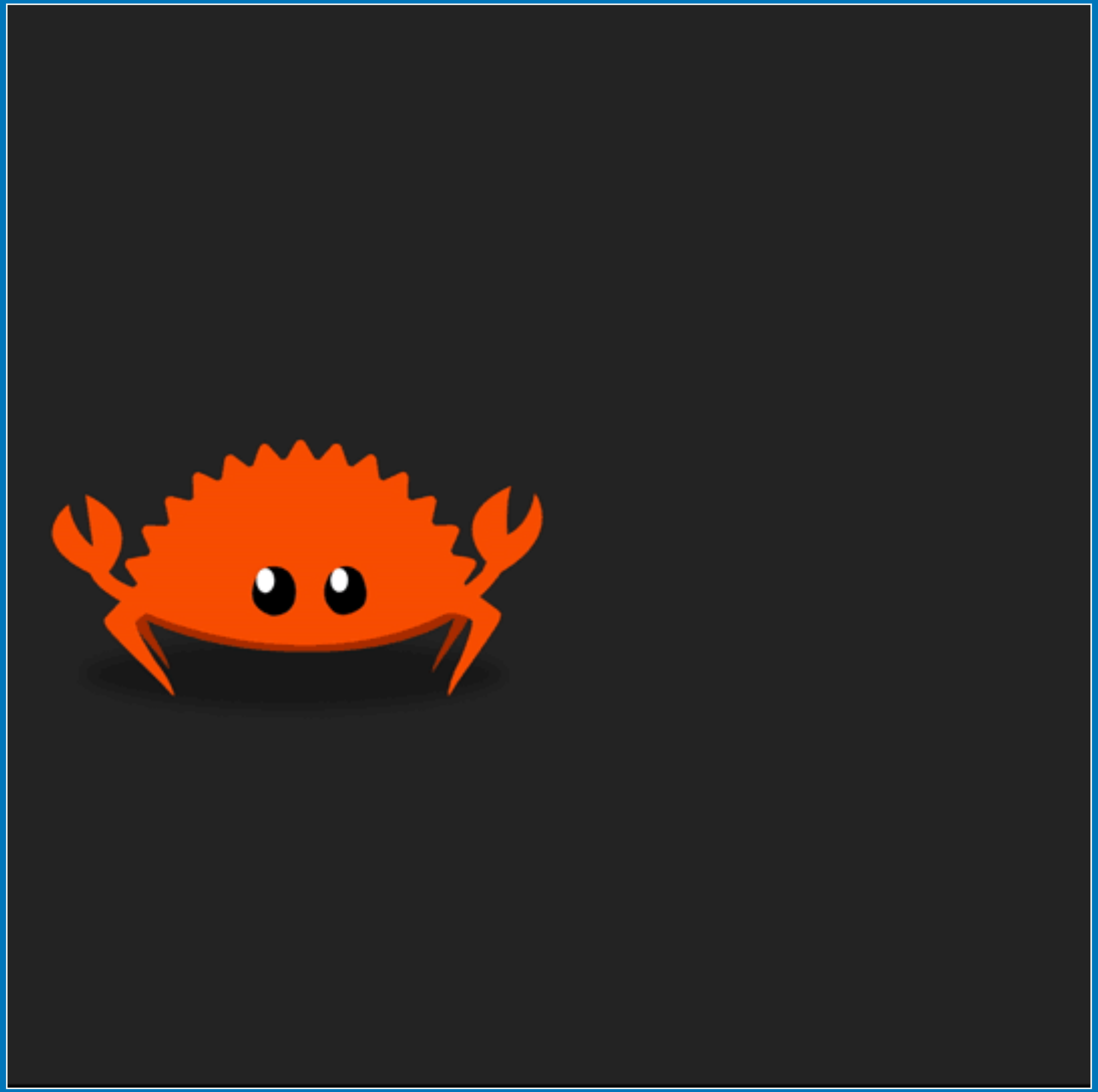
- Follows the WASI standard using wasmtime
- No networking
- More of a traditional "container" model of execution

Entertainment District

Engine Room

Why Rust?

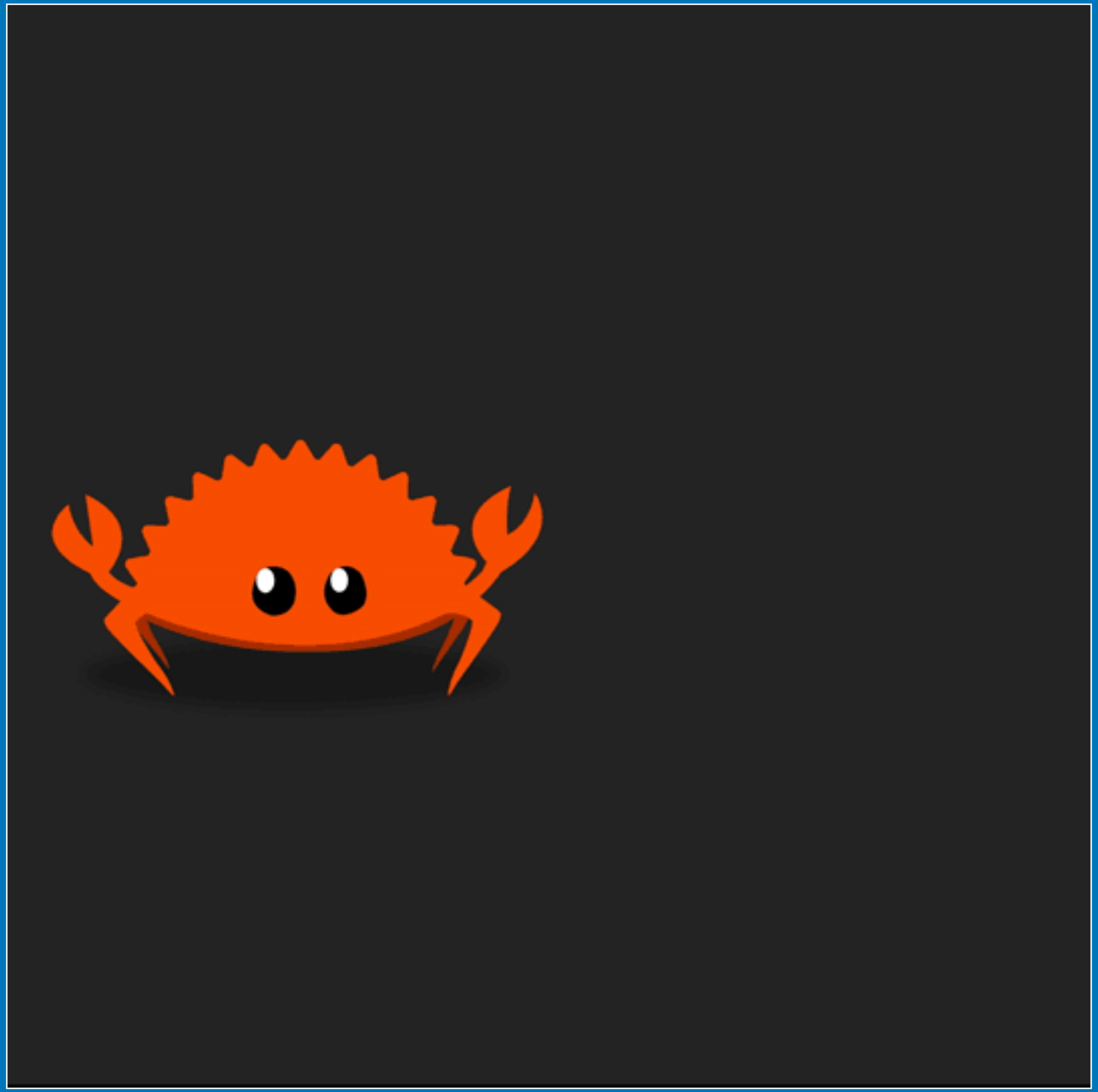
- WASM/WASI support
- Safety
- Extensibility
- Developer Experience



@_oftaylor

Why Rust?

- WASM/WASI support
- Safety
- Extensibility
- Developer Experience



@_oftaylor

Extensibility

```rust
let pod_client: Api<Pod> = Api::namespaced(client, "default");
```

```rust
let deploy_client: Api<Deployment> = Api::namespaced(client, "default");
```

```rust
impl<K> Api<K>
where
    K: k8s_openapi::Resource
```

```go
type PodInterface interface {
    Create(ctx context.Context, pod *v1.Pod, opts metav1.CreateOptions) (*v1.Pod, error)
    Update(ctx context.Context, pod *v1.Pod, opts metav1.UpdateOptions) (*v1.Pod, error)
    UpdateStatus(ctx context.Context, pod *v1.Pod, opts metav1.UpdateOptions) (*v1.Pod,
error)
    Delete(ctx context.Context, name string, opts metav1.DeleteOptions) error
    DeleteCollection(ctx context.Context, opts metav1.DeleteOptions, listOpts
metav1.ListOptions) error
    Get(ctx context.Context, name string, opts metav1.GetOptions) (*v1.Pod, error)
    List(ctx context.Context, opts metav1.ListOptions) (*v1.PodList, error)
    Watch(ctx context.Context, opts metav1.ListOptions) (watch.Interface, error)
    Patch(ctx context.Context, name string, pt types.PatchType, data []byte, opts
metav1.PatchOptions, subresources ...string) (result *v1.Pod, err error)
    GetEphemeralContainers(ctx context.Context, podName string, options metav1.GetOptions)
(*v1.EphemeralContainers, error)
    UpdateEphemeralContainers(ctx context.Context, podName string, ephemeralContainers
*v1.EphemeralContainers, opts metav1.UpdateOptions) (*v1.EphemeralContainers, error)

    PodExpansion
}
```

```go
type SecretInterface interface {
    Create(ctx context.Context, secret *v1.Secret, opts metav1.CreateOptions) (*v1.Secret,
error)
    Update(ctx context.Context, secret *v1.Secret, opts metav1.UpdateOptions) (*v1.Secret,
error)
    Delete(ctx context.Context, name string, opts metav1.DeleteOptions) error
    DeleteCollection(ctx context.Context, opts metav1.DeleteOptions, listOpts
metav1.ListOptions) error
    Get(ctx context.Context, name string, opts metav1.GetOptions) (*v1.Secret, error)
    List(ctx context.Context, opts metav1.ListOptions) (*v1.SecretList, error)
    Watch(ctx context.Context, opts metav1.ListOptions) (watch.Interface, error)
    Patch(ctx context.Context, name string, pt types.PatchType, data []byte, opts
metav1.PatchOptions, subresources ...string) (result *v1.Secret, err error)
    SecretExpansion
}
```

@_oftaylor

Extensibility

```rust
pub fn pod_key<N: AsRef<str>, T: AsRef<str>>(namespace: N, pod_name: T) -> String {
    format!("{}:{}", namespace.as_ref(), pod_name.as_ref())
}
```

@_oftaylor

Extensibility

```rust
pub fn pod_key<N: AsRef<str>, T: AsRef<str>>(namespace: N, pod_name: T) -> String {
    format!("{}:{}", namespace.as_ref(), pod_name.as_ref())
}
```

```go
func PodKey(namespace: fmt.Stringer, podName: fmt.Stringer) (string, error) {
  if namespace == nil || podName == nil {
    return nil, fmt.Errorf("namespace and pod name must not be nil")
  }
  return fmt.Sprintf("%s:%s", namespace, podName)
}
```

Extensibility

```rust
pub fn pod_key<N: AsRef<str>, T: AsRef<str>>(namespace: N, pod_name: T) -> String {
    format!("{}:{}", namespace.as_ref(), pod_name.as_ref())
}
```

```go
func PodKey(namespace: interface{}, podName: interface{}) (string, error) {
  if namespace == nil || podName == nil {
    return nil, fmt.Errorf("namespace and pod name must not be nil")
  }
  switch namespace := namespace.(type) {
  case string:
     os.Stdout.WriteString(v)
  case fmt.Stringer:
    namespace = namespace.String()
  default:
    return nil, fmt.Errorf("unknown type given: %T", namespace)
  }
  // ...
}
```

@_oftaylor

CRD Example

```rust
#[derive(CustomResource, Serialize, Deserialize, Default, Clone)]
#[kube(group = "clux.dev", version = "v1", namespaced)]
pub struct FooSpec {
    name: String,
    info: String,
}
```

```rust
println!("kind = {}", Foo::KIND);
let foos: Api<Foo> = Api::namespaced(client, "default");
let f = Foo::new("my-foo");
println!("foo: {:?}", f)
println!("crd: {}", serde_yaml::to_string(Foo::crd());
```

@_oftaylor

Developer Experience

Dependency Management

```
kube = "0.33.0"
k8s-openapi = { version = "0.7", default-features = false, features = ["v1_17"] }
```

```
#[cfg(any(feature = "cli", feature = "docs"))]
#[cfg_attr(feature = "docs", doc(cfg(feature = "cli")))]
pub struct Opts
```

@_oftaylor

Developer Experience

Ease of Coding

- Macros and metaprogramming
- Error handling
- Flow control (match blocks and unwrapping)

Caveats

- Rust Kubernetes Library missing some advanced features
- Async runtimes
- The logarithmic learning curve

Interested in helping?

https://github.com/deislabs/krustlet

- Documentation for GKE, Digital Ocean, IBM, etc.
- Feedback on issues
- Try it out and file bugs
- Joining our weekly call (link in project README)
- Good at Rust? Help us refactor
- Better ARM support

@_oftaylor

Questions?

@_oftaylor

References

- Krustlet: https://github.com/deislabs/krustlet
- waSCC: https://wascc.dev
- Some blog posts
  - High level overview/context: https://aka.ms/krustlet-overview
  - Intro blog: https://deislabs.io/posts/introducing-krustlet/
  - Kubernetes + Rust:
    - https://deislabs.io/posts/kubernetes-a-rusty-friendship/
    - https://msrc-blog.microsoft.com/2020/04/29/the-safety-boat-kubernetes-and-rust/

Thank You!

@_oftaylor