PlugFlow

WoSC10

Thomas Oberroither
University of Innsbruck

Philipp Gritsch
University of Innsbruck

Sashko Ristov
University of Innsbruck

Michael Felderer
German Aerospace Center (DLR)

December 2, 2024

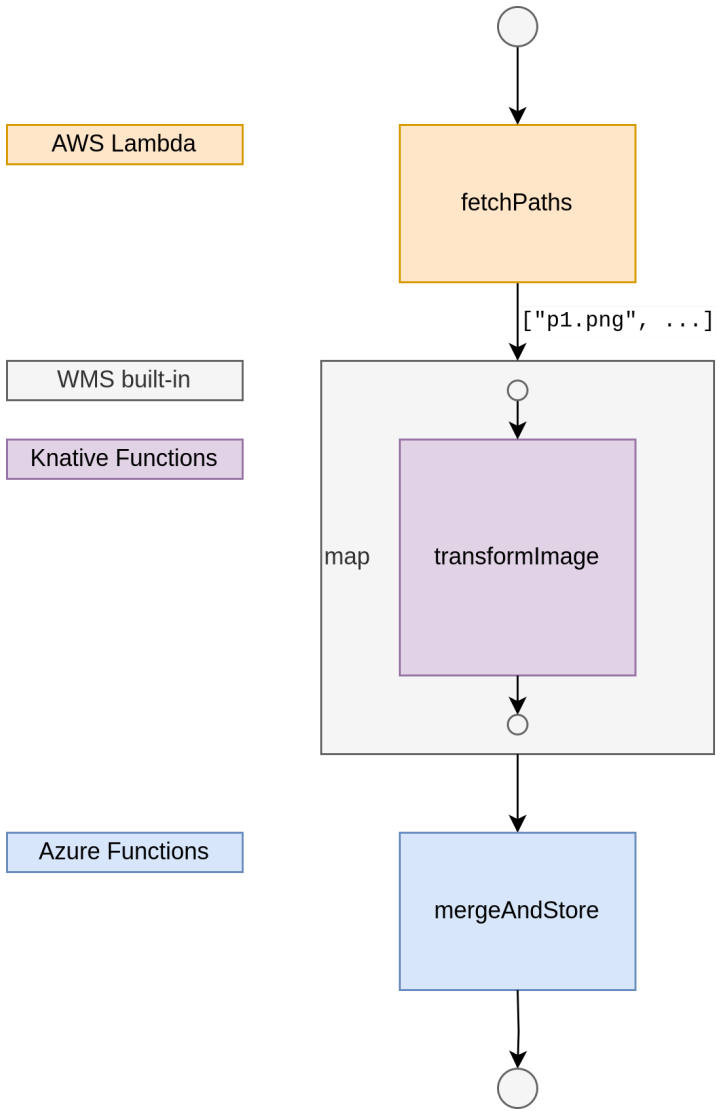
universität innsbruck

Workflows ...

Composed of serverless functions

- Written in different languages
- Deployed on different platforms
- Executed by a Workflow Management System (WMS)

Example: Map & Reduce



universität innsbruck

Common issues

- Small functions come with the same **overhead** as large ones
  - *Dependency management, deployment, etc.*
- **Flow options are limited** to what the WMS supports
  - *Conditional branching*
  - *Parallelism, loops, etc.*

What if we let workflow developers …

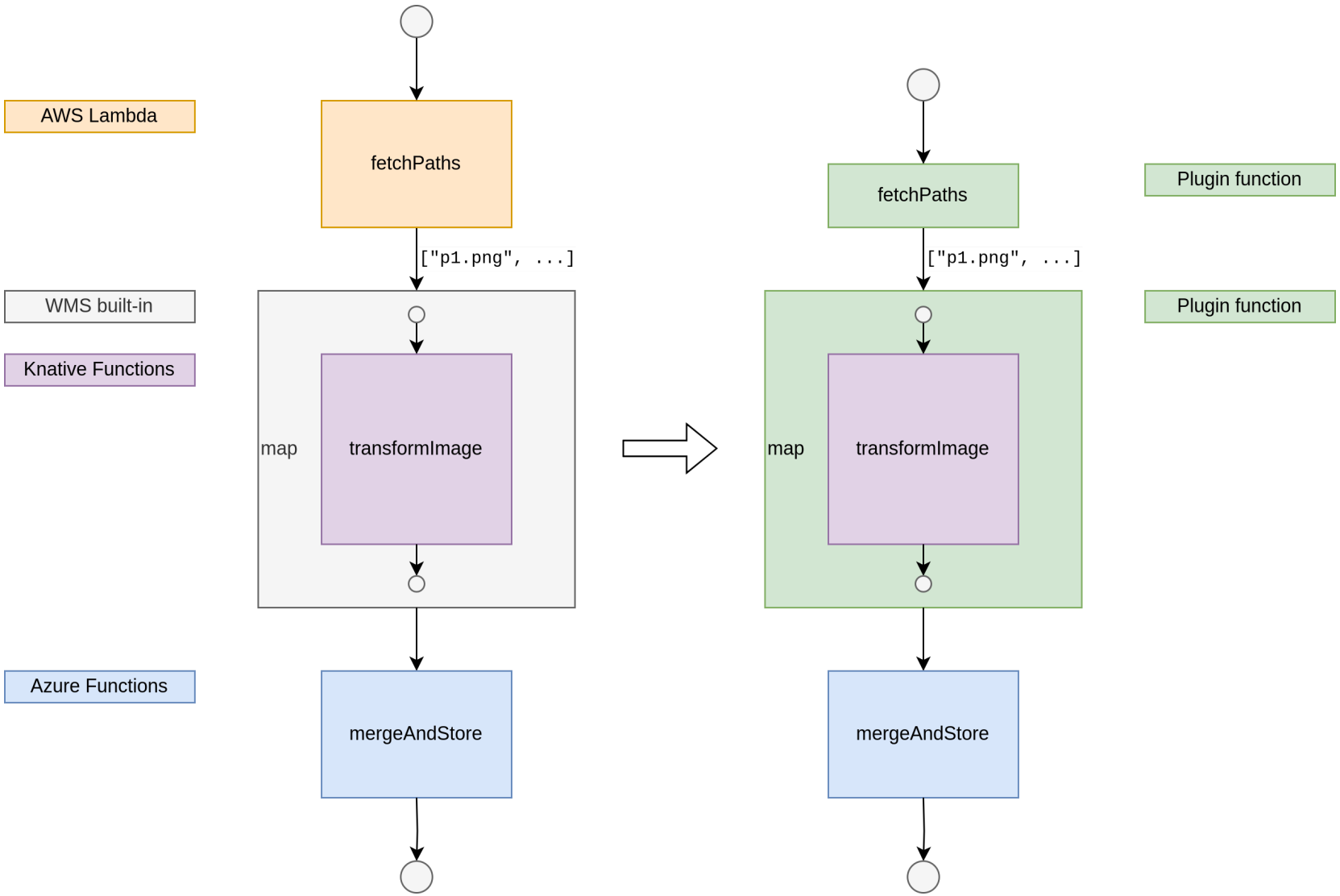
1. **Plug** small **functions** directly **into the engine**?
2. **Create custom control flow elements** using such functions?
   - *Next to out-of-the-box ones like* `map`, `parallel`, *etc.*
3. **Encode** workflow logic **within functions** themselves?
   - *Functions decide what's next*

PlugFlow

A hybrid approach to composing and managing workflows.

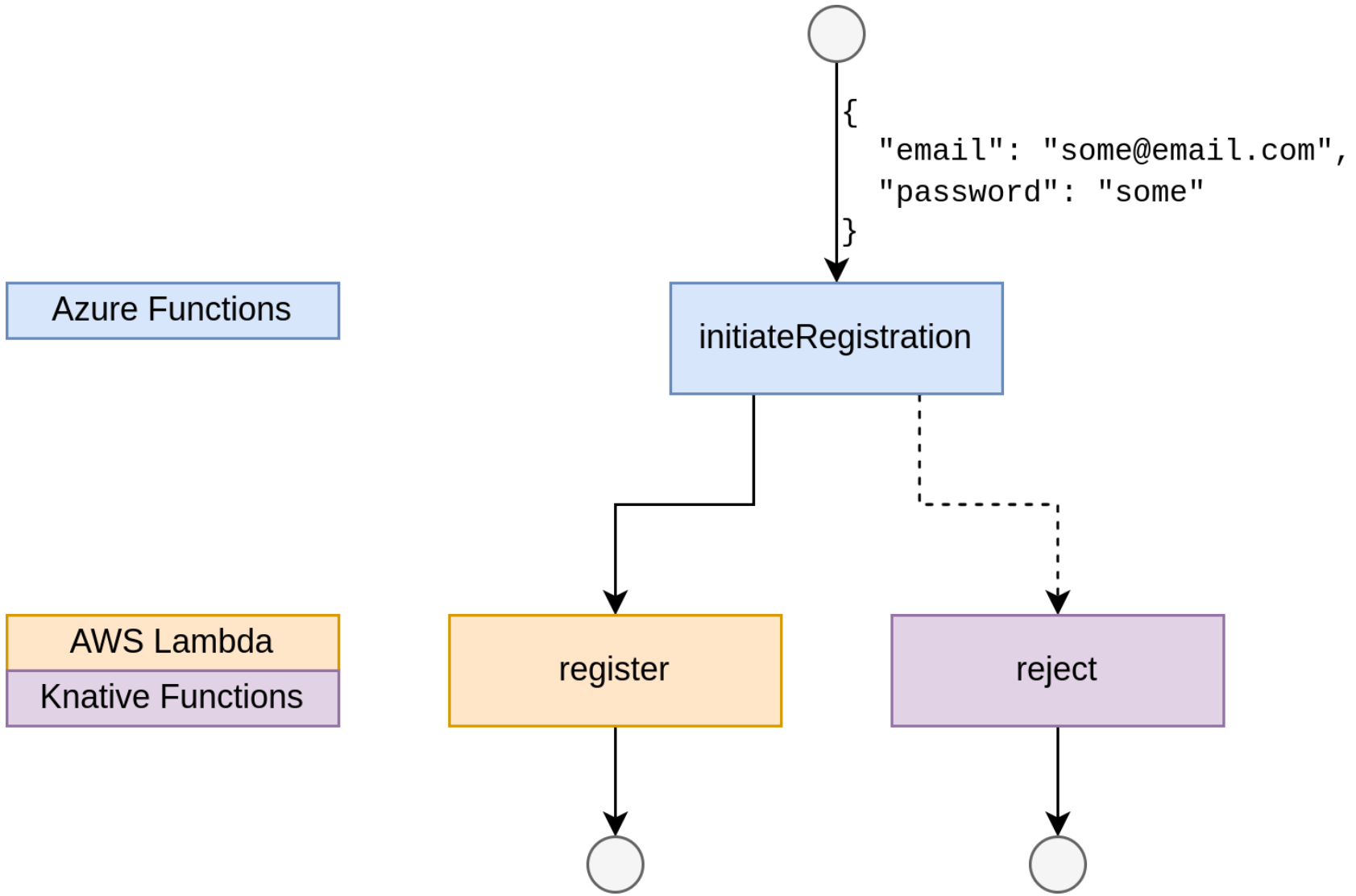
universität innsbruck

Using PlugFlow



universität innsbruck

Functions decide what's next



universität innsbruck

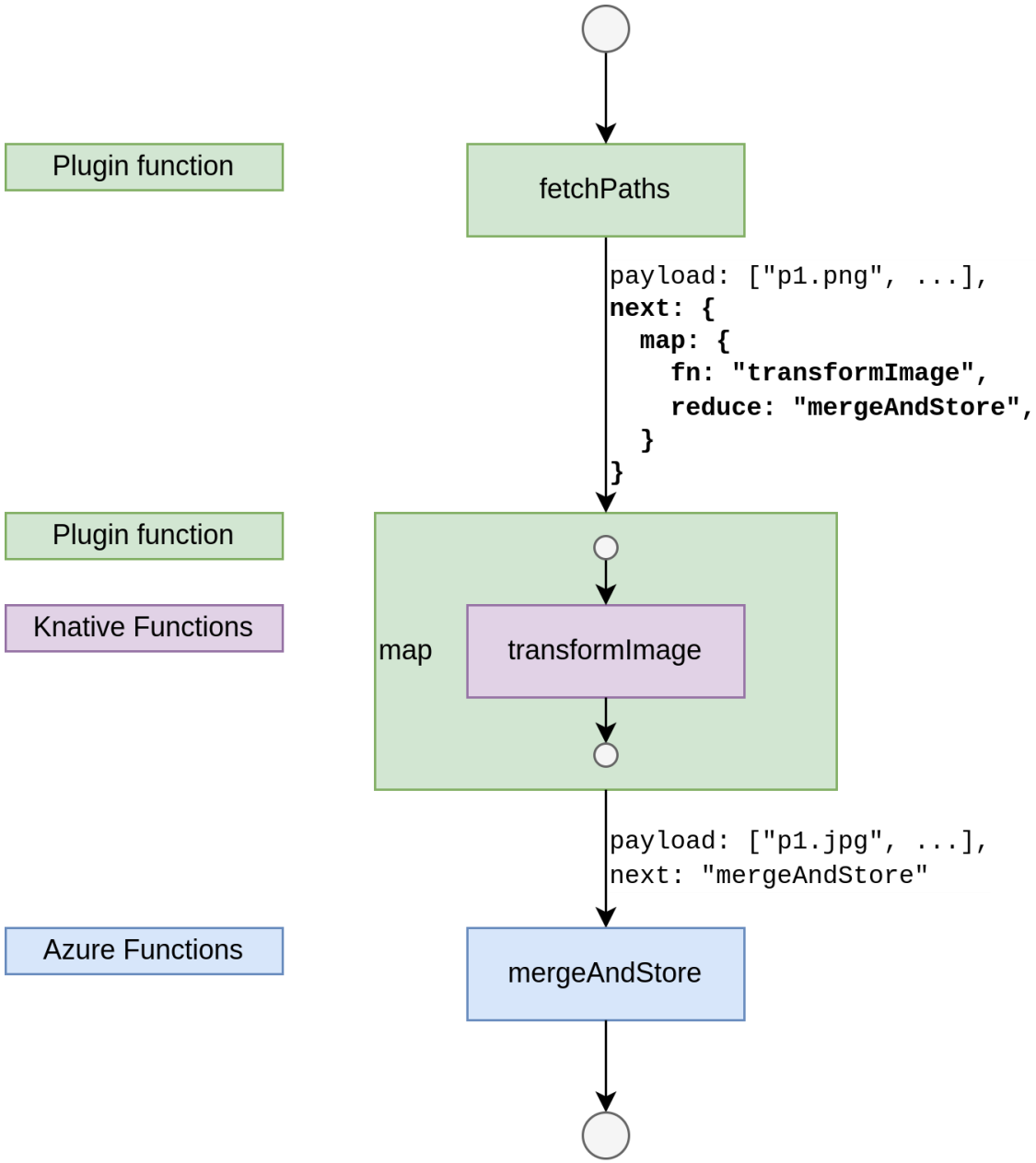
initiateRegistration:

```
export default async (requestBody: RequestBody): Promise<ResponseBody> => {

  const user = requestBody.payload;
  const email = user.email;
  const password = user.password;

  return {
    payload: user,
    next: !email.includes(password) ? "register" : "reject"
  };
};
```

Plugin functions as control flow elements



universität innsbruck

Plugin `map`

- Has **access to parameters** passed as part of the `next` field:

```
const { fn, reduce } = requestBody.context.params;
// -> { fn: "transformImage", reduce: "mergeAndStore" }
```

- **Invokes** `fn` for each element in the payload:

```
const promises = requestBody.payload.map((element) => {
  return context.run(fn, element);
});
```

- Will **forward** the result to the `reduce` function:

```
return {
  payload: await Promise.all(promises),
  next: reduce
};
```

universität innsbruck

Demo

Via CLI

Deploy:

```
cli/deploy.ts --workflow-directory workflows/demo --deployment-name demo
```

Run:

```
cli/run.ts --deployment-name demo --entrypoint fetchPaths --payload '{}'
```

Output:

```
Result: [ "p1.jpg", "p2.jpg" ]
Workflow: demo 1603ms
+ fetchPaths 9ms {}
| map 843ms ["p1.png","p2.png"]
  + transformImage 833ms "p1.png"
  + transformImage 825ms "p2.png"
| mergeAndStore 706ms ["p1.jpg","p2.jpg"]
```

Via Web UI

PF

Dashboard Deployments Runs

Run #48

Deployment: demo

Entrypoint: fetchPaths

Input:

```
{}
```

Output:

```
[
  "p1.jpg",
  "p2.jpg"
]
```

Stack:

```
Workflow: demo (12:23:40.393Z) (12:23:41.996Z) 1603ms
+ fetchPaths (12:23:40.416Z) (12:23:40.425Z) 9ms {}
| map (12:23:40.429Z) (12:23:41.272Z) 843ms ["p1.png","p2.png"]
  + transformImage (12:23:40.434Z) (12:23:41.267Z) 833ms "p1.png"
  + transformImage (12:23:40.437Z) (12:23:41.262Z) 825ms "p2.png"
| mergeAndStore (12:23:41.277Z) (12:23:41.983Z) 706ms ["p1.jpg","p2.jpg"]
```

universität innsbruck

Thank you!

universität
innsbruck