

Drift

an imperative programming
environment for the cloud

#2 Implementation

Overview


- Drift Language

- Concepts
- Examples

- Drift Execution

- Drift FS
- Architecture
- Error Model

Recap

- Want: ‘language of the system’
- Workflow languages one possible domain
 - black box tasks, ...
- Bash: system-view coordination
 - black box tasks, immediate feedback, ...
 - Problem: FS → shared mutable state
- Functional  Distributed
 - Cuneiform: functional, distributed, ...

Recap

Bash	Functional
&	- (lazy)
	function composition
>, >>	name binding
<	-
\$	eval

Drift Language

Can we build an *imperative, stateful, interpreted* language for distributed (micro) service coordination ?

- Need ‘state’ to be stateful on
- Essence: data + services

“Do this on that thing over here.”

“Now do this on that and put it over there”

Drift Language

- How do we (humans) ‘interact’ with data?
 - need ‘names’ to identify and retrieve our data
 - need ‘names’ to give data *meaning*
- *Names* are at the center of programming!
 - only names and services
- Names best be hierarchical → Namespaces
 - Names, Namespaces and Services