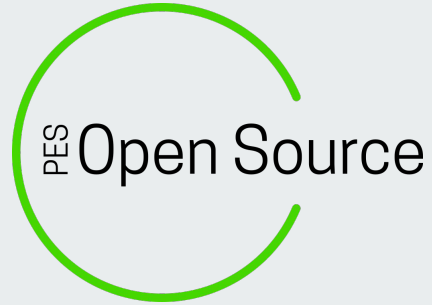




Imperative vs Declarative Systems

Madhav Jivrajani
Ft. Kuby





Introductions

- I'm Madhav (@maddyoiiii on slack)
- We have another friend helping us out today

Introductions

- This is Kuby
- A little bit about Kuby:
 - Loves walks and treats
 - Hates baths
 - Unaware of heart-melting potential





Outline for today

- What are these things called automata?
- “Model” of “Computation”
- Imperative systems
- Declarative systems
- Demo!
- Rewind; relating today’s topics to Kubernetes (vot dat)

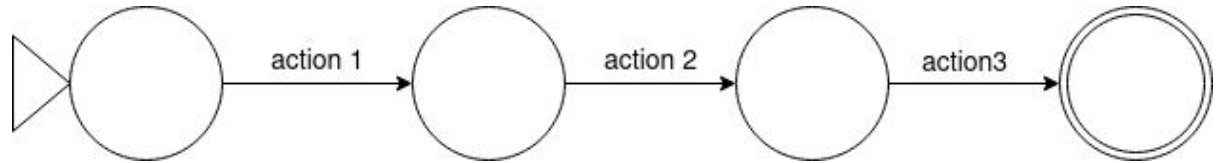


Auto...what?

- States
 - Finite
- Transitions b/w states
 - Actions
- Start state
- Final state(s)

Auto...what?

- States
 - Finite
- Transitions b/w states
 - Actions
- Start state
- Final state(s)





Some lingo

- Desired state
- Mitigating actions
- State match
- State drift

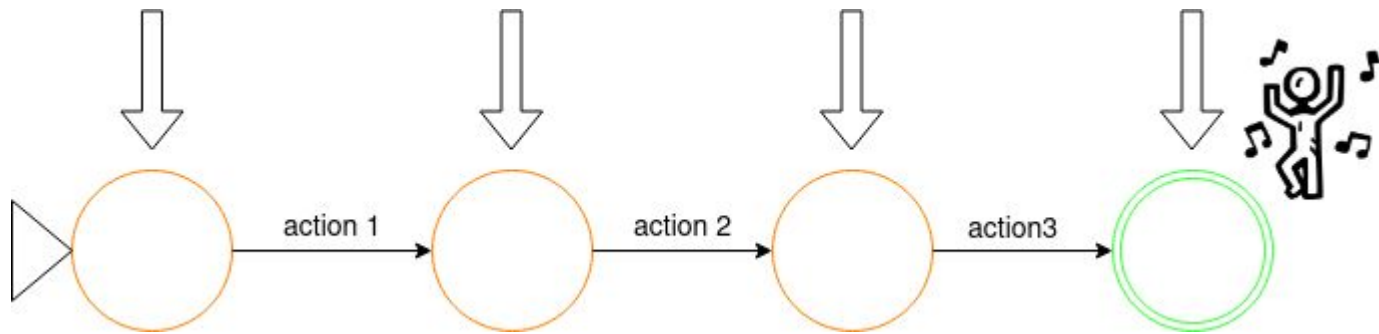
Some lingo

- Desired state
 - Where we would like to be
- Mitigating actions
 - Actions leading to desired state
- State match
 - Current state = desired state
- State drift
 - Current state \neq desired state

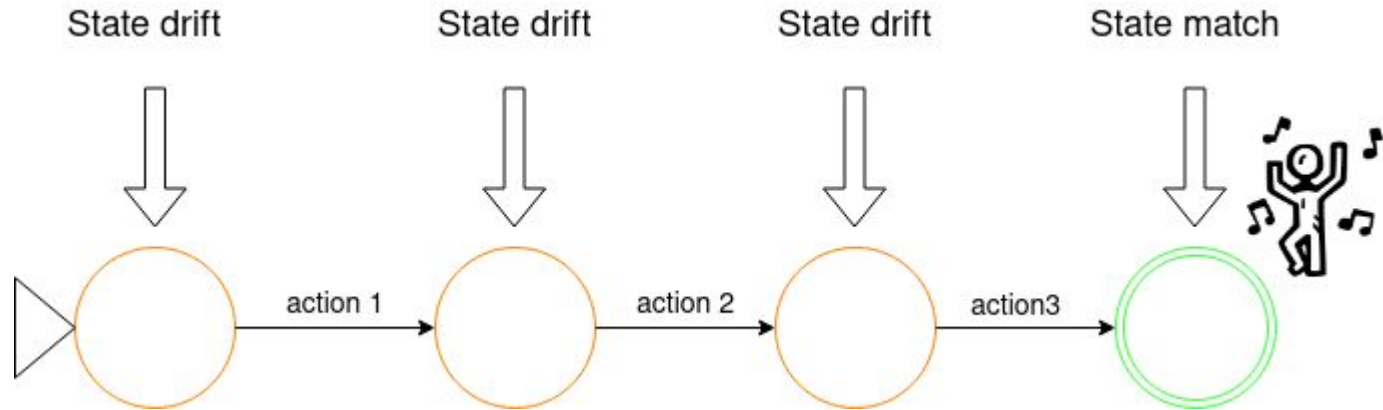
“U GOT THISH” ~ Kubey, 2021



Ohhhh, *automata*!



Ohhhh, *automata*!



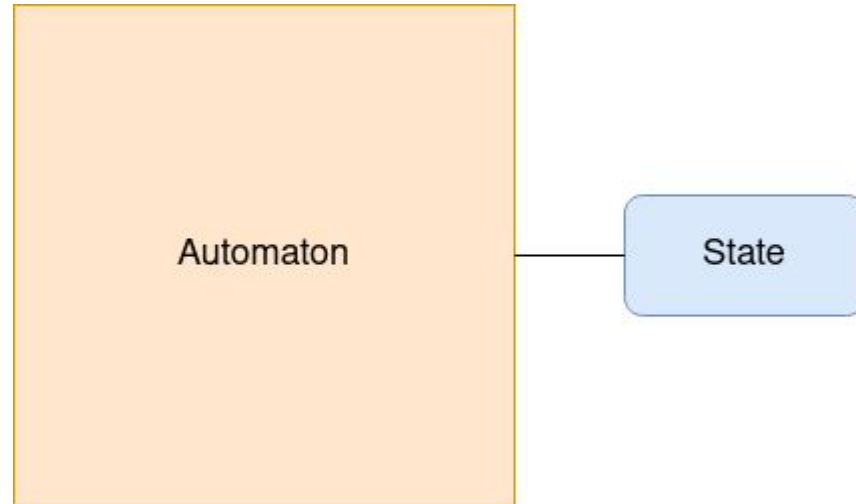


“Model” of “Computation”

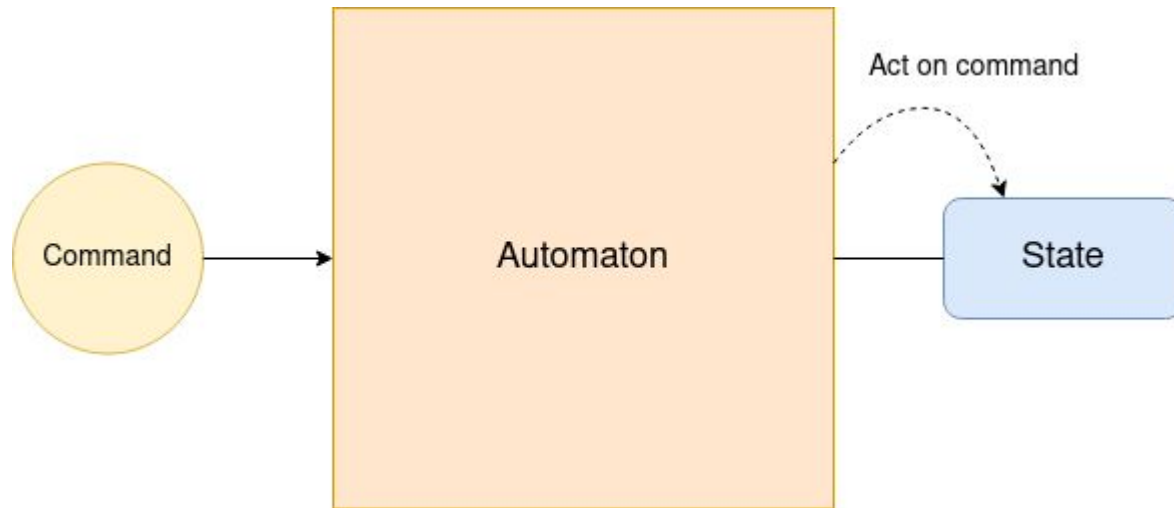
- Model?
- of
 - lol
- Computation?



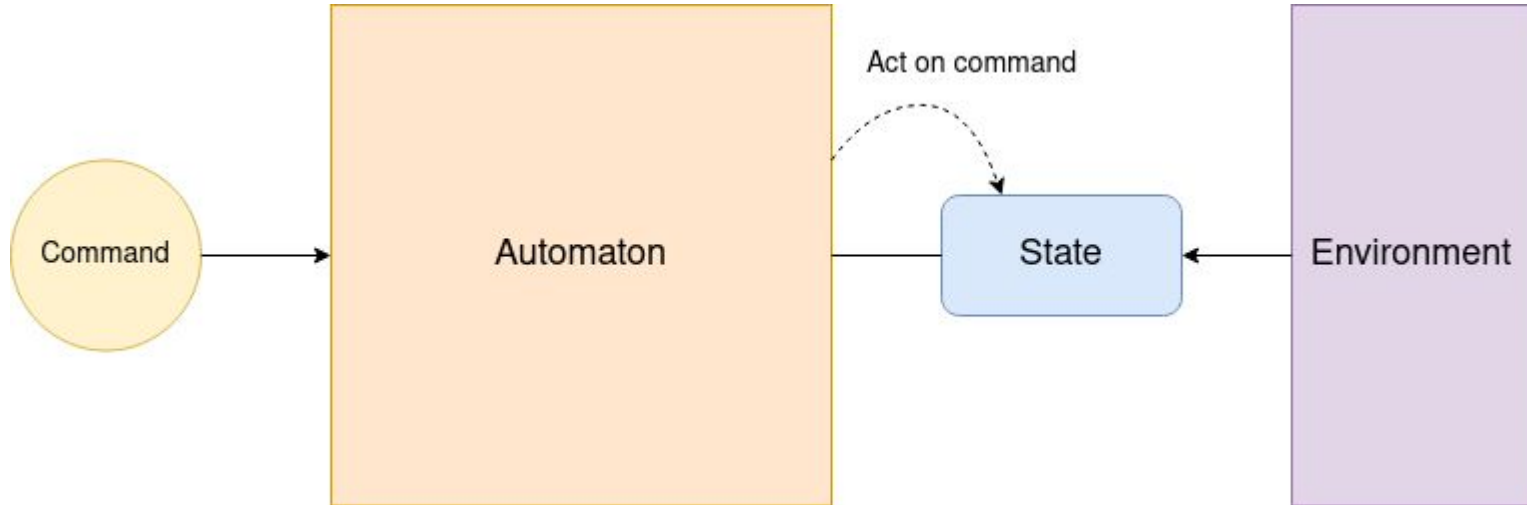
Model of Computation



Model of Computation



Model of Computation



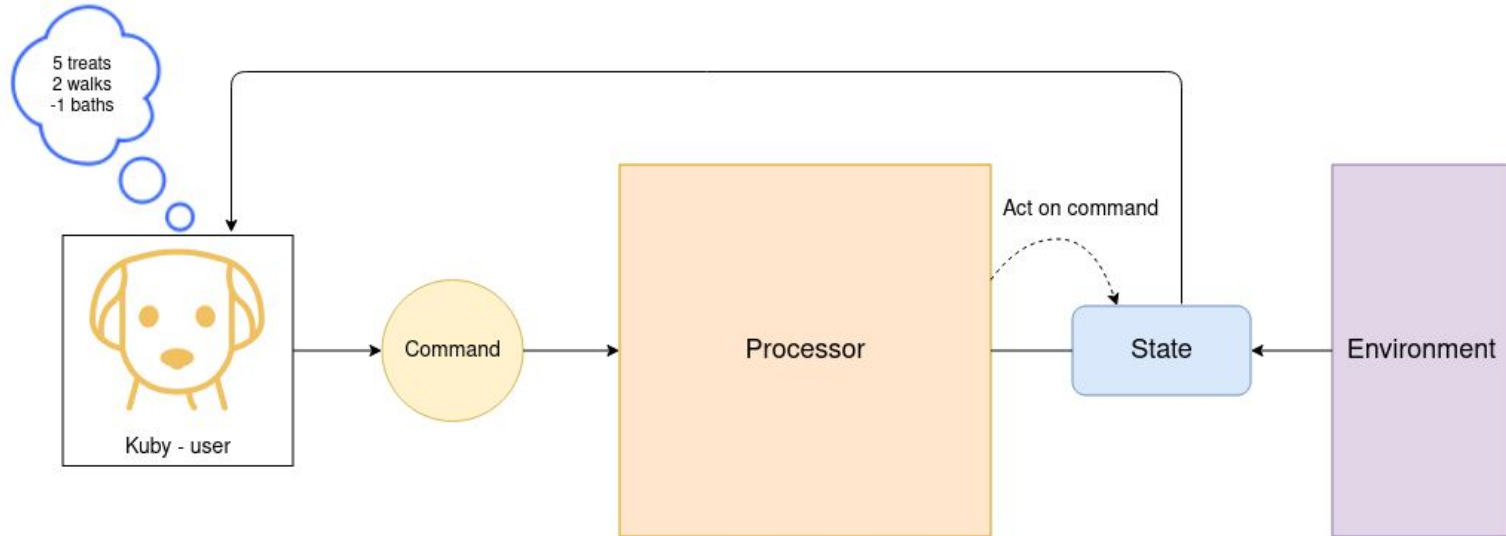


Imperative Systems

- User knows the desired state.
- User provides “commands” to achieve desired state.
 - Mitigating actions.
- Commands executed by the *processor*.
- What does this look like?

Imperative Systems

Desired state





Imperative Systems

When would you use them?

- Number of unintended state changes is low.
 - For such changes that *do* end up occurring, the user should be able to provide commands to drive the system towards the desired state.
- When you need stricter control over execution of commands.
- When your system does need “self-healing” type of capabilities
 - More on this later

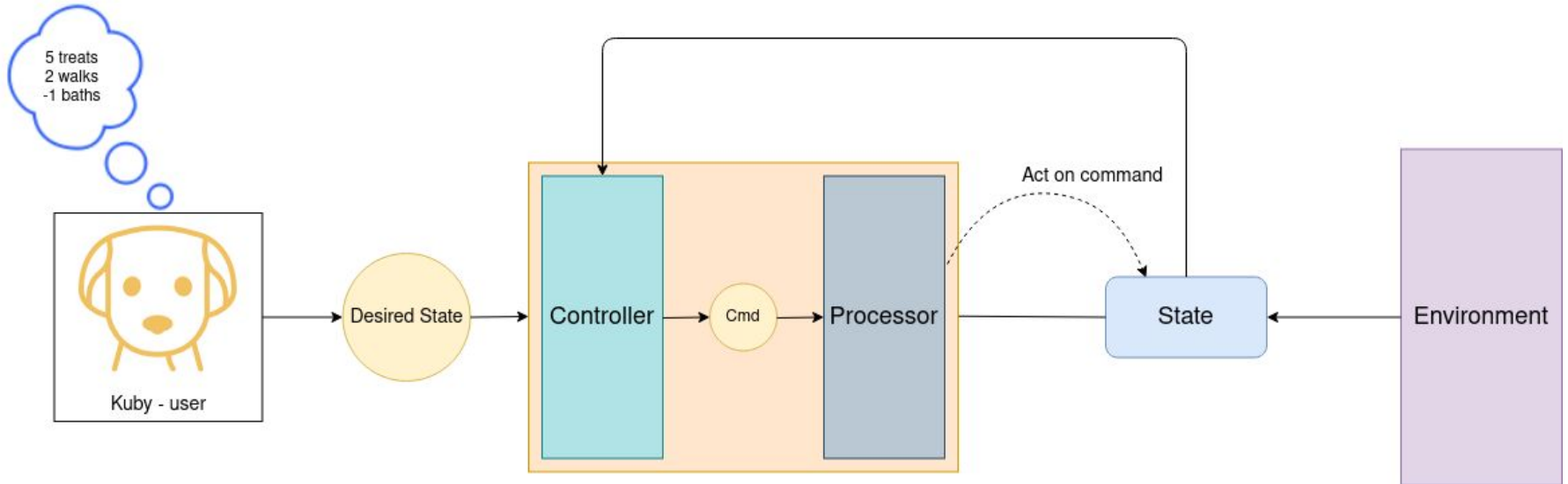


Declarative Systems

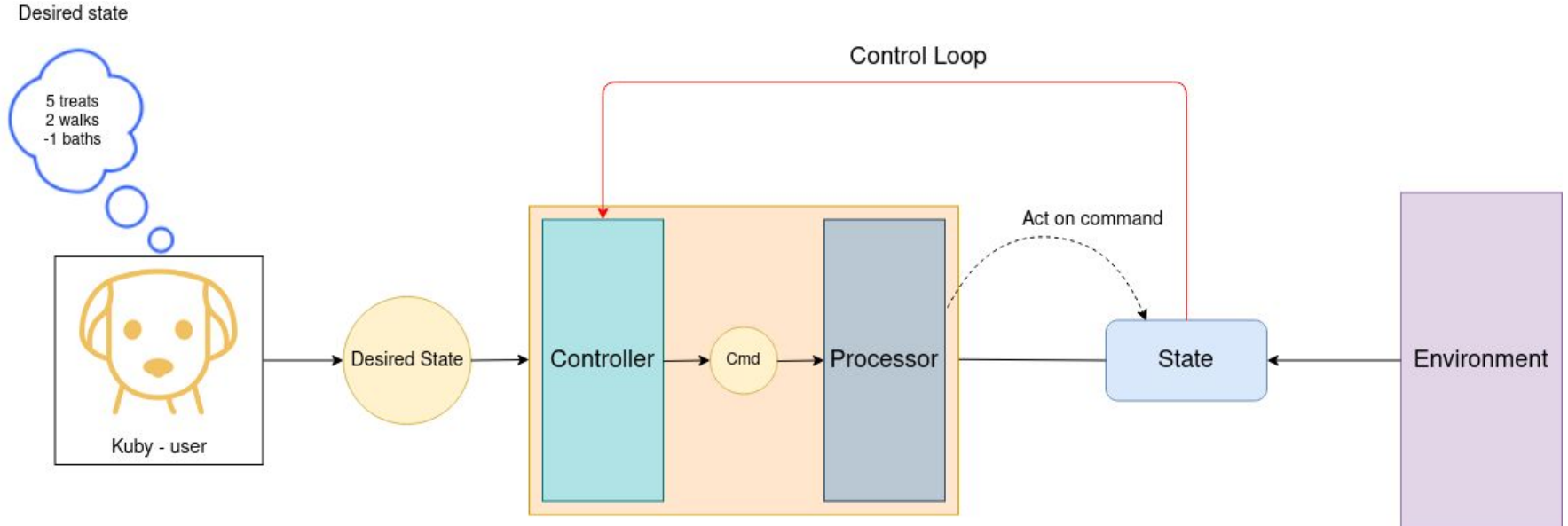
- User knows the desired state.
- User provides the desired state to the system.
- The system observes the current state of the system and *issues* commands to move towards the desired state.
 - Component of the system that does this is the *controller*.
- Component of the system that executes these instructions is the *processor*.
- What does this look like?

Declarative Systems

Desired state



Declarative Systems





Demo!

- multiprocessing
 - Process
- flask
 - GET
- PIDs
- Reconciling



Kubernetes and declarative systems

- Desired state: config files
- Controller
 - Control loops
- State
 - etcd
- Processor
 - Kubernetes API



References

- This whole presentation was inspired by this amazing blog post by *Dominik Tornow*:
<https://dominik-tornow.medium.com/imperative-vs-declarative-8abc7dcae82e>
- Kubernetes documentation for controllers:
<https://kubernetes.io/docs/concepts/architecture/controller/>



Slides and source

- Slides: <http://bit.ly/pesos-ivds>
- Source code: <https://github.com/MadhavJivrajani/pesos-imperative-declarative>



Questions?



Thank you!

“Thankoooo” ~ Kubey, 2021

