LOCAL VARIABLES MEETUP FEB. 18 2016

AWS SIMPLE WORKFLOW

SO WHO IS THIS GUY, ANYWAY?

CHRIS@HLPRMNKY.COM

- Software developer for ~17 years
  - *Mostly* Java, some Ruby and Python, Obj-C and Swift too
    - Primarily concerned with backend data processing, having the devops
- Last ~2 years have used AWS SWF in anger every day
- Current project processes ~2.5k records, day, will ramp up to ~20k/day soon, ~1million/day soon after that

AWS SIMPLE WORKFLOW – THE ONE-SLIDE SUMMARY

- Distributed workflow coordination
- Think Onyx, Celery, etc. but let AWS manage all the stuff Jepsen would yell at you about if you built it yourself
- Units of work can live anywhere (on-premise, cloud, mobile devices, anything that can poll an HTTP endpoint)
- AWS manages handing out work and passing results to your workflow logic
- AWS maintains a correct, authoritative history of decisions and work done for each workflow execution

SIMPLE WORKFLOW GLOSSARY

- **Decider**: Code that schedules *activities* to gather information and do work; business logic lives here
- **Activity**: Code that completes a unit of work: fetches a piece of configuration data, writes a completed file, records data, sends an email, etc.
- **Execution**: A single run of decider and activity tasks from the scheduling of a “start workflow” decider task until the decider schedules a “complete workflow”/“fail workflow” or a timeout limit is reached

SIMPLE WORKFLOW GLOSSARY 2/2

- **Task List**: SWF-managed "queue" of work. *Activities* pull a unit of work off their task list; *Deciders* pull the result of an activity *plus the workflow execution history* off theirs
- **Domain**: An SWF domain organizes decider and activity type registrations and workflow executions, for example grouped by dev/stage/prod or blue/green, etc.
  Note that SWF Domains are not cross-region

BUSINESS CASE

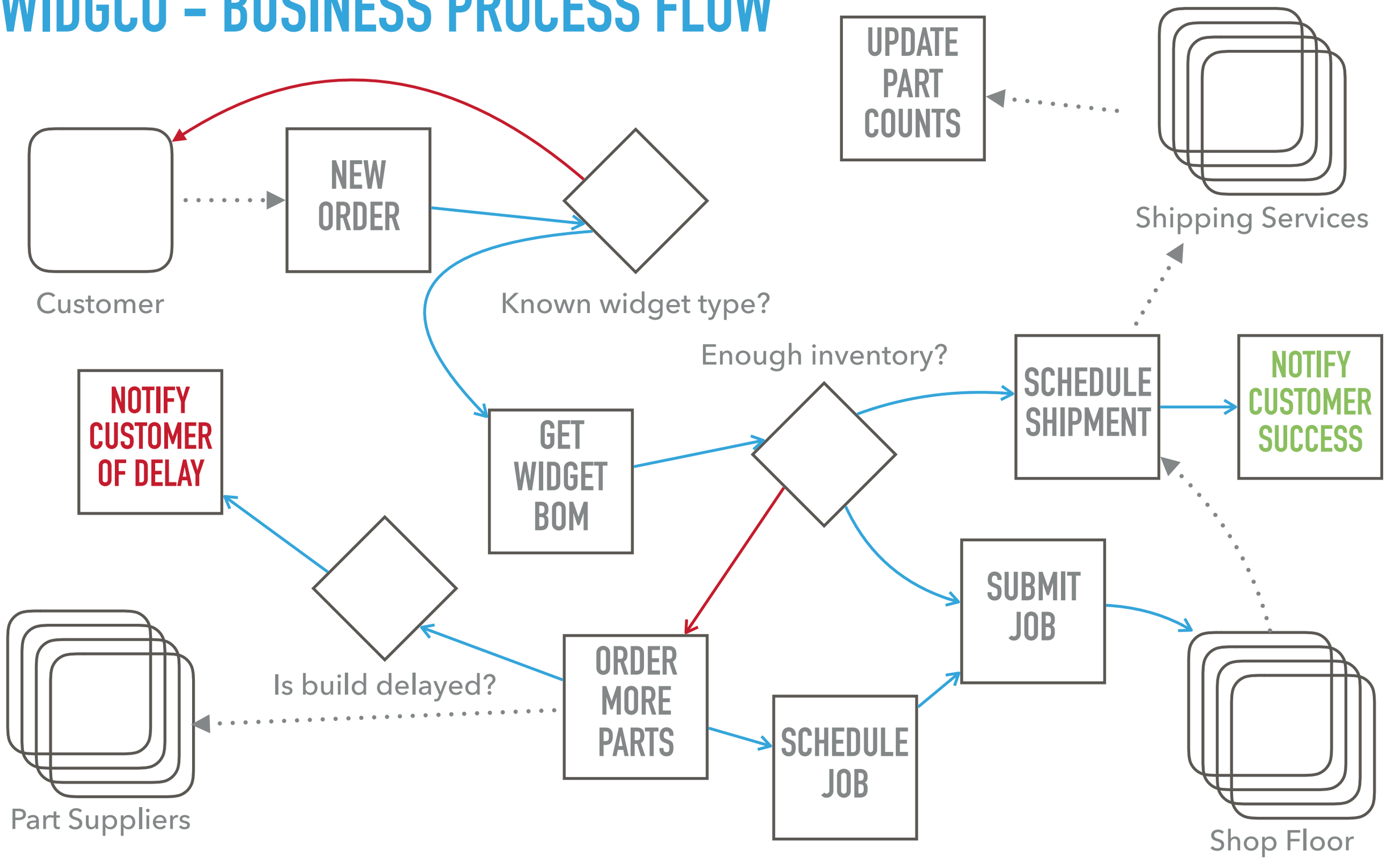
WIDGCO

"Widgets on Demand"

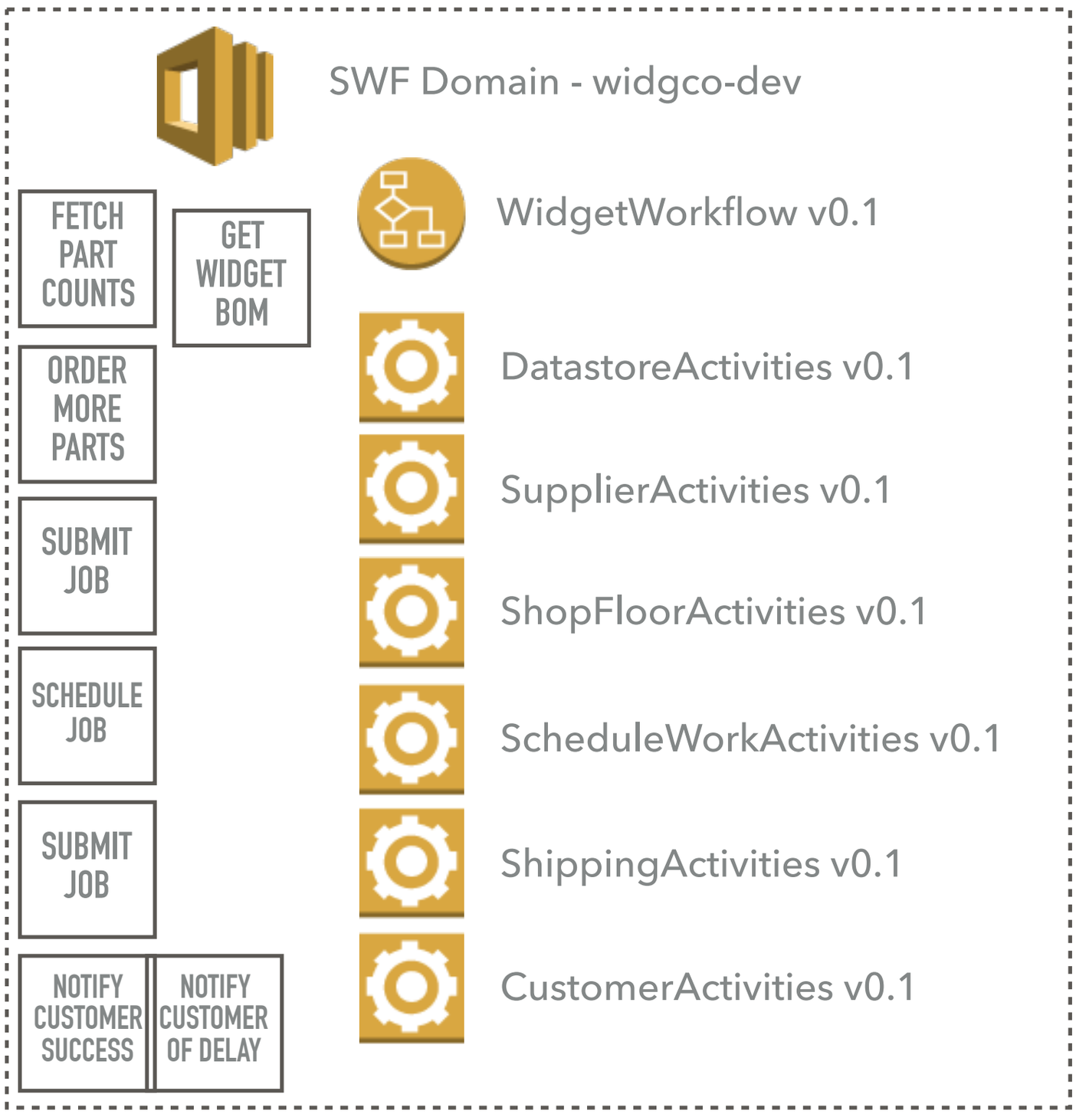
WIDGCO – LEGACY INFRASTRUCTURE

- 100% on-premise
- Business logic in monolithic app
- Data stored in single DB instance
- Communication to vendors, shippers via SOAP, EDI, maybe some REST on the horizon
- Scalability is the largest challenge facing Widgco today

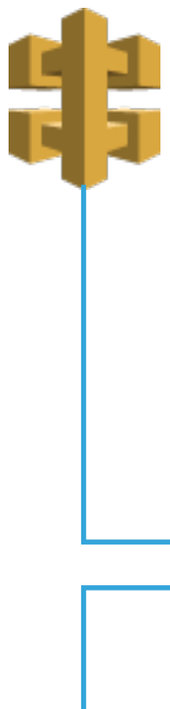
WIDGCO – BUSINESS PROCESS FLOW



WIDGCO – SWF DESIGN



SIMPLE WORKFLOW – INCOMING ORDERS



REST endpoint /submitOrder

Workflow starter - Lambda function builds input JSON and submits a startWorkflow decision task to widgco-decider task list

Decider processes task input, schedules an activity task on widgco-datastore tasklist to get widget bill of materials

SIMPLE WORKFLOW – INVENTORY MANAGEMENT

Decider receives a decision task with the current part counts for the parts in the current BOM

There aren't enough of part SKU "Thing1" available for this job to complete

Decider schedules two activity tasks:

SupplierActivities.orderMoreParts

CustomerActivities.notifyDelayedOrder

Possible enhancement - get part ETA from orderMoreParts and work that into an estimate to the customer in notifyDelayedOrder

SIMPLE WORKFLOW – MANUAL CONFIRMATION

Decider schedules an activity task on the ScheduleWorkflowActivities taskList

ScheduleWorkflowActivities publishes a workflow token and stays in “running” state

TIME PASSES

After the new inventory arrives and work can proceed, human worker finds and resumes the workflow token with an API call from a desktop, tablet, phone, etc.

Decider receives a decision task with the completed schedule job task and schedules a SubmitJob activity task on the ShopFloorActivities taskList

LET'S SEE IT WORK

SAY HELLO TO

WIDG.CO

"Widgets as a Service"

SWF FLOW FRAMEWORK

- “Extra” SDK that provides significant additional functionality
- Kinesis SDK : K(C|P)L :: SWF SDK : Flow
- Only available for Java and Ruby
  - Java uses compile-time *source* weaving, so really “Java” not “JVM language you want to use”
  - Ruby SDK lagging behind Java in features

SWF FLOW FRAMEWORK FOR JAVA

- Deciders become “Workflows” - imperative, synchronous style instead of FSM
- All annotations all the time
- Activities and Workflows require an interface; changing the interface contracts requires bumping the Activity/Workflow version
- Asynchrony handled via a Promise abstraction
- Workflow code can be called *many times* as different async calls to activity tasks are realized
- Activity code remains synchronous and stateful

SWF FLOW FRAMEWORK – TESTING

- Java SDK has a JUnit class runner
  - builds workflow with stubbed activity implementations that can mock exceptions, store call inputs
  - allows for running the virtual clock at different rates
- Both Ruby and Java have a workflow *replayed*
  - initialize with a workflow execution history
  - re-run, inject mocks, set breakpoints, whatever
- Both systems run locally and don't incur usage charges

SWF VISIBILITY WITH AWS CLOUDWATCH

- Metrics available can be helpful but are not always sufficient
  - Can monitor start-to-close time but no insight as to why it goes up/down without e.g. EC2 CPU or network latency information
- Useful for profiling a workflow, spotting trends
- *Very* useful for identifying AWS throttling limits before workflow executions start to fail

GENERAL THOUGHTS ON GETTING THE DEVOPS

- All the SWF code you deploy is just services
  - Logstash -> Sumologic your logs
  - EC2 workers in ASGs can scale on CPU, incoming queue depth, other Cloudwatch metrics
- CloudFormation doesn't have SWF resources, so your SWF domains need to be created "by hand"
- Activity and Workflow versions register themselves, task lists register themselves - useful for blue/green testing and rolling deploys

QUESTIONS?

RESOURCES

- https://github.com/hlprmnky/swf-tech-talk/
- AWS SWF docs
- chris@hlprmnky.com
- Image credit:
  - Sewing machine by Viphu Vac http://streetwill.co/posts/436-sewing-machine01
  - Windmill provided by Keynote

THANKS!