Cloud Foundryの死活監視

@Kuruma

本スライドは下記URIで公開しています:

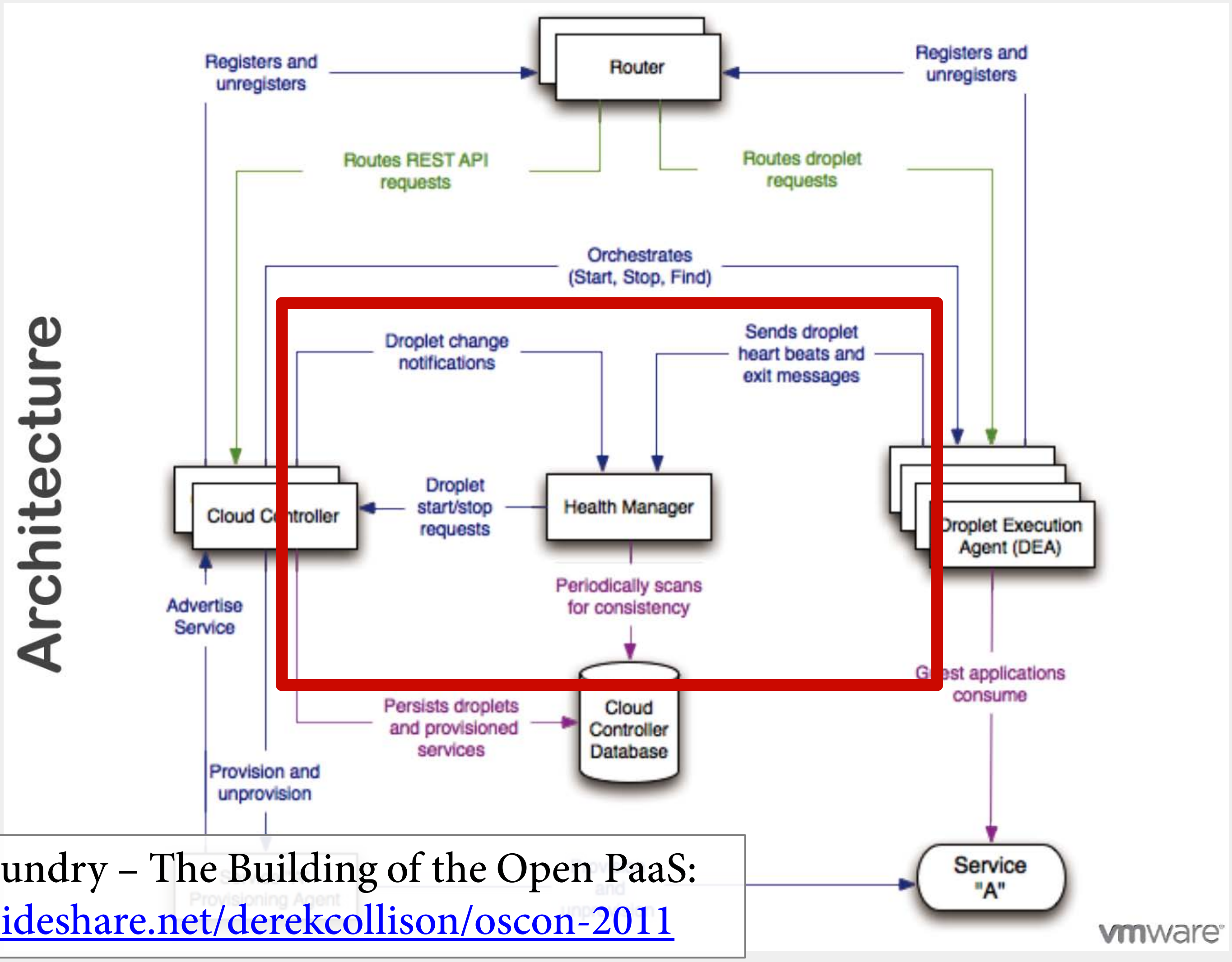
http://bit.ly/cfcrjp03-kuruma

http://kuruman.org/diary/2011/12/15/file/cloud-foundry-reading_kuruma.pdf

# whoami

- Kuruma
  - Web Accessibilityに始まり下へ下へ
  - twitter.com/Kuruma
  - kuruman.org

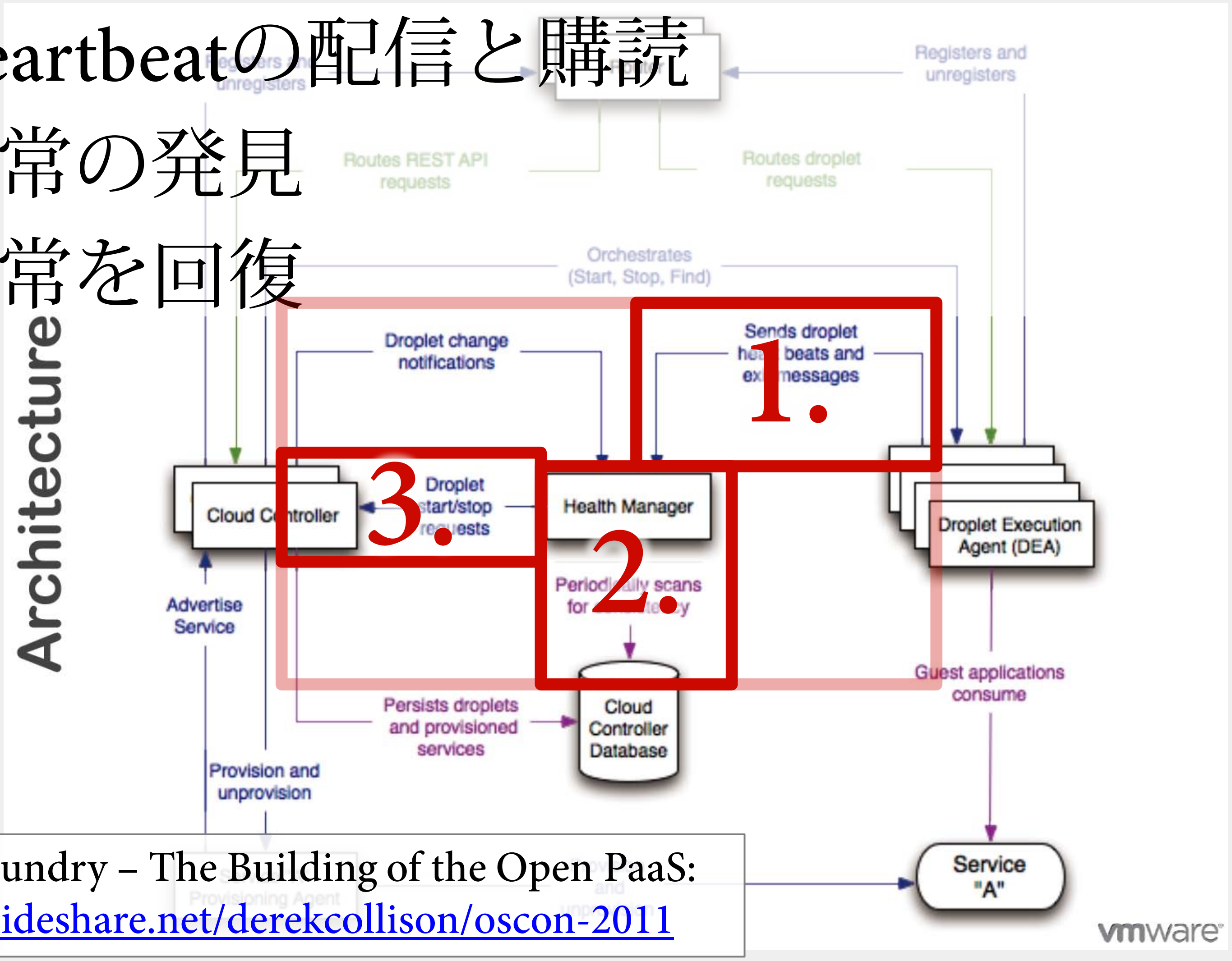
今回の範囲


Architecture
Router
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
Orchestrates (Start, Stop, Find)
Droplet change notifications
Sends droplet heart beats and exit messages
Cloud Controller
Droplet start/stop requests
Health Manager
Droplet Execution Agent (DEA)
Advertise Service
Periodically scans for consistency
Guest applications consume
Persists droplets and provisioned services
Cloud Controller Database
Provision and unprovision
Provisioning Agent
Service "A"
vmware

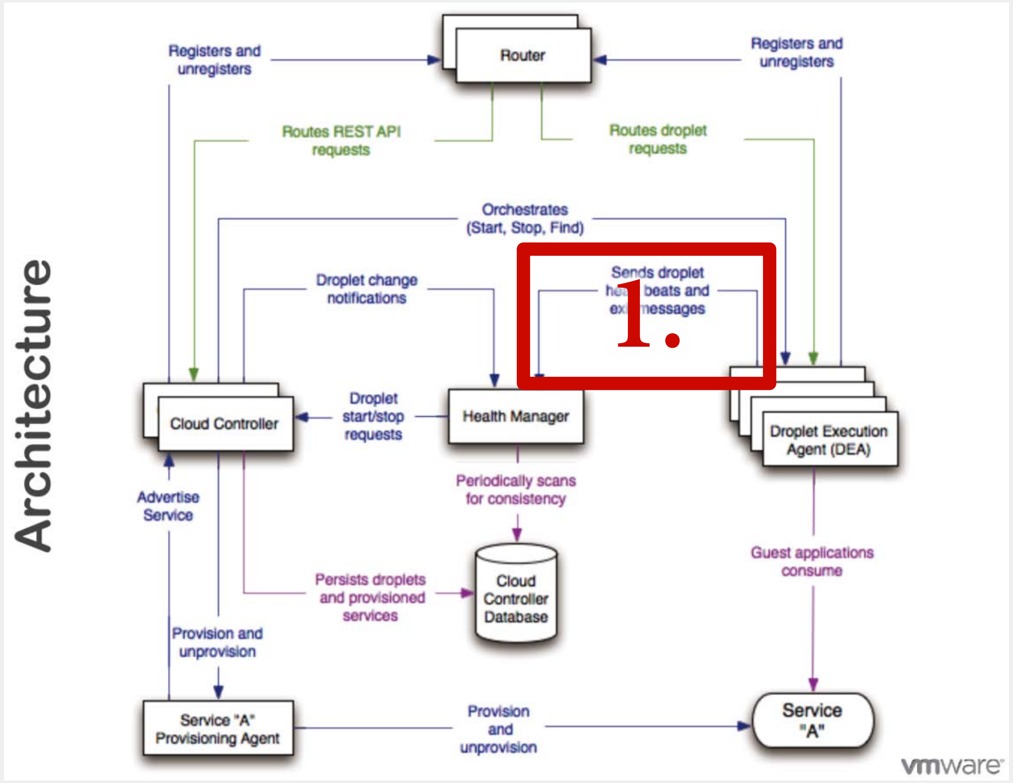
via. Cloud Foundry – The Building of the Open PaaS: http://www.slideshare.net/derekcollison/oscon-2011

概観

1. Heartbeatの配信と購読
2. 異常の発見
3. 異常を回復


Architecture
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
Orchestrates (Start, Stop, Find)
Droplet change notifications
Sends droplet heart beats and exit messages
1.
Cloud Controller
3.
Droplet start/stop requests
Health Manager
2.
Periodically scans for consistency
Droplet Execution Agent (DEA)
Advertise Service
Guest applications consume
Persists droplets and provisioned services
Cloud Controller Database
Provision and unprovision
Provisioning Agent
Service "A"
vmware

via. Cloud Foundry – The Building of the Open PaaS:
http://www.slideshare.net/derekcollison/oscon-2011


Architecture
Router
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
Orchestrates (Start, Stop, Find)
Droplet change notifications
Sends droplet heartbeats and exit messages
1.
Cloud Controller
Droplet start/stop requests
Health Manager
Droplet Execution Agent (DEA)
Advertise Service
Periodically scans for consistency
Guest applications consume
Persists droplets and provisioned services
Cloud Controller Database
Provision and unprovision
Service "A" Provisioning Agent
Provision and unprovision
Service "A"
vmware

@DEA

1-1. HEARTBEATの配信

DEAが定期的に状態を配信

- Heartbeatの配信
- アプリケーションの動作状況確認

```
MONITOR_INTERVAL = 2 # 2 secs
CRASHES_REAPER_INTERVAL = 30 # 30 secs
def initialize(config)
  ……
  @heartbeat_interval = config[‘intervals’][‘heartbeat’] #= 10
  ……
end
def run()
  ……
  EM.add_periodic_timer(@heartbeat_interval)  { send_heartbeat }
  EM.add_timer(MONITOR_INTERVAL) { monitor_apps }
  EM.add_periodic_timer(CRASHES_REAPER_INTERVAL) { crashes_reaper }
  ……
end
```

vcap/dea/lib/dea/agent.rb

Heartbeatの配信

- 何かインスタンスが動作している時
- 動作するインスタンスそれぞれについて
- Heartbeatを生成して
- publish

```
def send_heartbeat
  return if @droplets.empty? || @shutting_down
  heartbeat = {:droplets => []}
  @droplets.each_value do |instances|
    instances.each_value do |instance|
      heartbeat[:droplets] << generate_heartbeat(instance)
    end
  end
  NATS.publish('dea.heartbeat', heartbeat.to_json)
end
```

vcap/dea/lib/dea/agent.rb

Heartbeat (Hash)

- 状態 (Symbol)
  - :STARTING, :RUNNING, :CRASHED, :STOPPED, :SHUTTING_DOWN
- 状態の最終更新時刻 (Time#to_i)

```ruby
def generate_heartbeat(instance)
  {
    :droplet => instance[:droplet_id],
    :version => instance[:version],
    :instance => instance[:instance_id],
    :index => instance[:instance_index],
    :state => instance[:state],
    :state_timestamp => instance[:state_timestamp]
  }
end
```

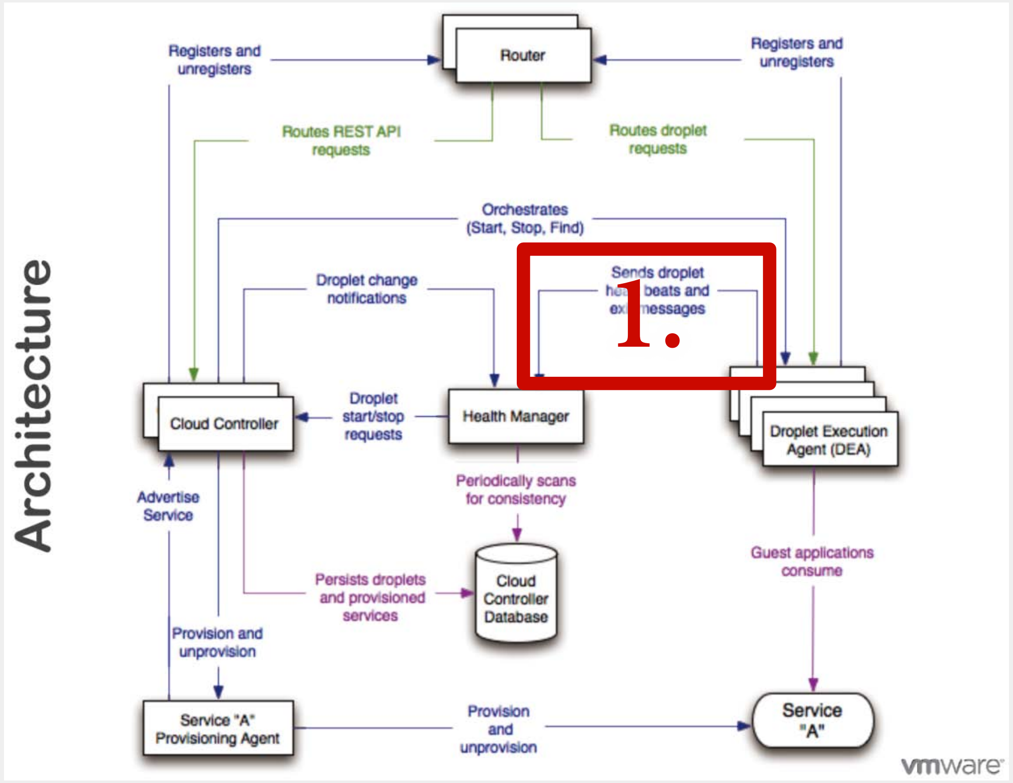
vcap/dea/lib/dea/agent.rb

アプリケーションの動作状況確認

- シェルを叩くだけ (ps, du)

```
def monitor_apps(startup_check = false)
  ……
  process_statuses = `ps axo pid=,ppid=,pcpu=,rss=,user=`.split("¥n")
  ……
    du_proc = proc do |p|
      p.send_data("cd #{@apps_dir}¥n")
      p.send_data("du -sk * 2> /dev/null¥n")
      p.send_data("exit¥n")
    end
    cont_proc = proc do |output, status|
      monitor_apps_helper(startup_check, start, du_start, output,
pid_info, user_info)
    end
    EM.system('/bin/sh', du_proc, cont_proc)
  ……
end
```

vcap/dea/lib/dea/agent.rb


Architecture
Router
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
Orchestrates (Start, Stop, Find)
Droplet change notifications
Sends droplet heartbeats and exit messages
1.
Cloud Controller
Droplet start/stop requests
Health Manager
Droplet Execution Agent (DEA)
Advertise Service
Periodically scans for consistency
Guest applications consume
Persists droplets and provisioned services
Cloud Controller Database
Provision and unprovision
Service "A" Provisioning Agent
Provision and unprovision
Service "A"
vmware

@HealthManager

1-2. HEARTBEATの購読

Heartbeatの購読

```
def run
  ……
  NATS.start(:uri => @config['mbus']) do
    ……
    subscribe_to_messages
  end
end
```

```
def subscribe_to_messages
  ……
  NATS.subscribe('dea.heartbeat') do |message|
      @logger.debug("heartbeat: #{message}")
      process_heartbeat_message(message)
  end
  ……
end
```

vcap/health_manager/lib/health_manager.rb

Heartbeat受信時の処理

- 状態等に応じて5通りに分岐

```
def process_heartbeat_message(message)
  ……
  parse_json(message)['droplets'].each do |heartbeat|
  droplet_id = heartbeat['droplet']
  instance = heartbeat['instance']
  droplet_entry = @droplets[droplet_id]
  if droplet_entry
    ……
    state = heartbeat['state']
    if RUNNING_STATES.include?(state)
      ……
    elsif state == CRASHED
      ……
    end
  else
    ……
  end
end
```

vcap/health_manager/lib/health_manager.rb

HealthManagerに於ける状態定義等

- APP_STABLE_STATE (droplet)
- RUNNING_STATE (instance)
- RESTART_REASONS (instance)

```ruby
# TODO - Oh these need comments so badly..
DOWN                  = 'DOWN'
STARTED               = 'STARTED'
STOPPED               = 'STOPPED'
CRASHED               = 'CRASHED'
STARTING              = 'STARTING'
RUNNING               = 'RUNNING'
FLAPPING              = 'FLAPPING'
DEA_SHUTDOWN          = 'DEA_SHUTDOWN'
DEA_EVACUATION        = 'DEA_EVACUATION'
APP_STABLE_STATES = Set.new([STARTED, STOPPED])
RUNNING_STATES        = Set.new([STARTING, RUNNING])
RESTART_REASONS       = Set.new([CRASHED, DEA_SHUTDOWN, DEA_EVACUATION])
```

vcap/health_manager/lib/health_manager.rb

1. droplet_entry存在せず

- HealthManagerが知らないインスタンス停止

```
def process_heartbeat_message(message)
  ……
  if droplet_entry
    ……
  else
    instance_uptime = Time.now.to_i - heartbeat['state_timestamp']
    health_manager_uptime = Time.now.to_i - @started
    threshold = @database_scan * 2

    if health_manager_uptime > threshold
           && instance_uptime > threshold
      @logger.info("Stopping unknown app: #{droplet_id}/#{instance}.")
      stop_instances(droplet_id, [instance])
    end
  end
end
```

vcap/health_manager/lib/health_manager.rb

2. 意図せぬインスタンスが起動中

- 当該インスタンスを終了
  - 起動しすぎた時、等
  - stop_instancesについては後述

```
def process_heartbeat_message(message)
  ……
    if RUNNING_STATES.include?(state)
      ……
      if index_entry[:state] == RUNNING
             && index_entry[:instance] != instance
        stop_instances(droplet_id, [instance])
      else
        ……
      end
    elsif state == CRASHED
  ……
end
```

vcap/health_manager/lib/health_manager.rb

3. 意図通り起動中または起動済

- タイムスタンプを更新
  - flippingを検知する際等に使用

```
def process_heartbeat_message(message)
  ……
    if RUNNING_STATES.include?(state)
      ……
      if index_entry[:state] == RUNNING
              && index_entry[:instance] != instance
        ……
      else
        index_entry[:instance] = instance
        index_entry[:timestamp] = Time.now.to_i
        index_entry[:state] = state.to_s
        index_entry[:state_timestamp] = heartbeat['state_timestamp']
      end
    elsif state == CRASHED
  ……
end
```

vcap/health_manager/lib/health_manager.rb

4. CRASHED

- クラッシュしたインスタンスとして格納
  - droplet_entry = @droplets[droplet_id]
- 一定時間以上経過後に削除

```
def process_heartbeat_message(message)
  ……
    if RUNNING_STATES.include?(state)
        ……
    elsif state == CRASHED
      droplet_entry[:crashes][instance] = {
        :timestamp => Time.now.to_i,
        :crash_timestamp => heartbeat['state_timestamp']
      }
    end
  ……
end
```

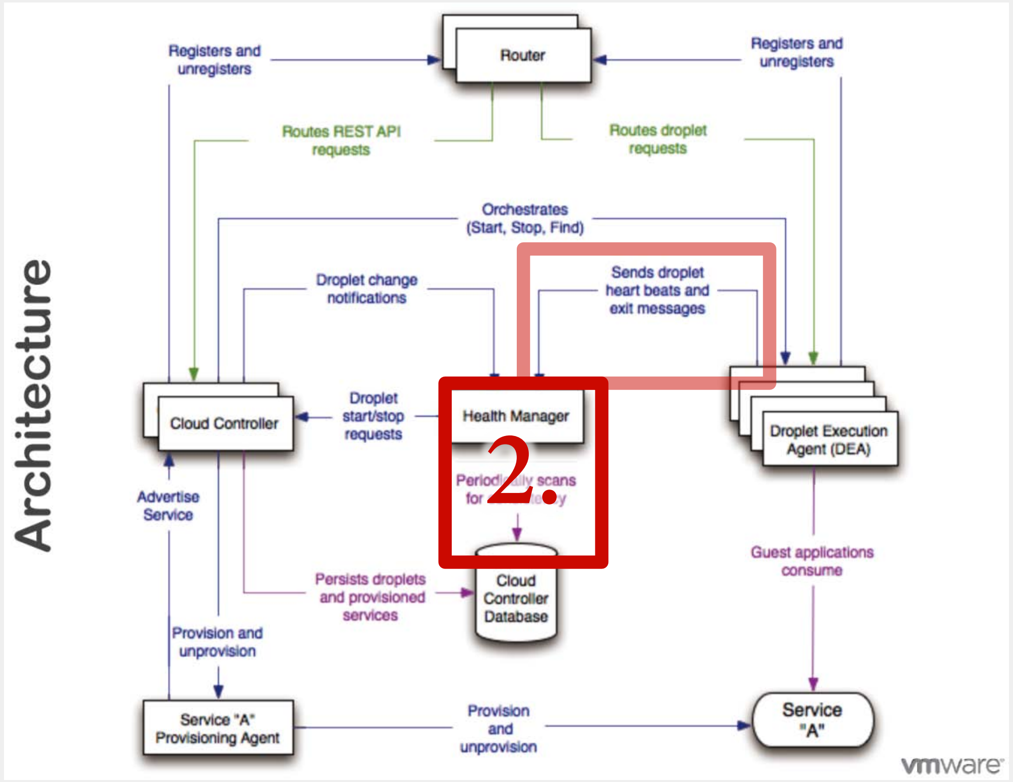
vcap/health_manager/lib/health_manager.rb

5. それ以外の状態

- 何もしない

```
def process_heartbeat_message(message)
  ……
  if droplet_entry
    ……
    state = heartbeat['state']
    if RUNNING_STATES.include?(state)
      ……
    elsif state == CRASHED
      ……
    end
  else
    ……
  end
end
```

vcap/health_manager/lib/health_manager.rb


Architecture
Router
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
Orchestrates (Start, Stop, Find)
Droplet change notifications
Sends droplet heart beats and exit messages
Cloud Controller
Droplet start/stop requests
Health Manager
Droplet Execution Agent (DEA)
2.
Periodically scans for
Advertise Service
Persists droplets and provisioned services
Cloud Controller Database
Guest applications consume
Provision and unprovision
Service "A" Provisioning Agent
Provision and unprovision
Service "A"
vmware

@DEA

2. 異常の発見

HealthManagerのバッチ処理

- DB情報取得、アプリケーション分析と記録

```
def run
  ……
    configure_timers
  ……
end
```

```
def configure_timers
  EM.next_tick { update_from_db }
  EM.add_periodic_timer(@database_scan) { update_from_db }
  ……
  EM.next_tick { analyze_all_apps(collect_stats = false) }
  ……
  EM.add_timer(@droplet_lost) do
    EM.add_periodic_timer(@droplets_analysis) { analyze_all_apps }
  end
  ……
end
```

vcap/health_manager/lib/health_manager.rb

アプリケーションの状態把握

- 初回は個々のアプリケーションを見ない
  - collect_stats: 初回はfalse

```
def analyze_all_apps(collect_stats = true)
  ……
  instances = crashed = 0
  stats =
      {:running => 0, :down => 0, :frameworks => {}, :runtimes => {}}
  @droplets.each do |id, droplet_entry|
    analyze_app(id, droplet_entry, stats) if collect_stats
    instances += droplet_entry[:instances]
    crashed += droplet_entry[:crashes].size if droplet_entry[:crashes]
  end
  VCAP::Component.varz[:total_apps] = @droplets.size
  VCAP::Component.varz[:total_instances] = instances
  VCAP::Component.varz[:crashed_instances] = crashed
  ……
end
```

vcap/health_manager/lib/health_manager.rb

個々のアプリケーション状態記録

```
def analyze_all_apps(collect_stats = true)
  ……
  @droplets.each do |id, droplet_entry|
    analyze_app(id, droplet_entry, stats) if collect_stats
    ……
  end
  ……
  if collect_stats
    VCAP::Component.varz[:running_instances] = stats[:running]
    VCAP::Component.varz[:down_instances] = stats[:down]
    VCAP::Component.varz[:running][:frameworks] = stats[:frameworks]
    VCAP::Component.varz[:running][:runtimes] = stats[:runtimes]
    @logger.info("Analyzed #{stats[:running]} running and #{stats[:down]} down apps in #{elapsed_time_in_ms(start)}")
  else
    @logger.info("Analyzed #{@droplets.size} apps in #{elapsed_time_in_ms(start)}")
  end
end
```

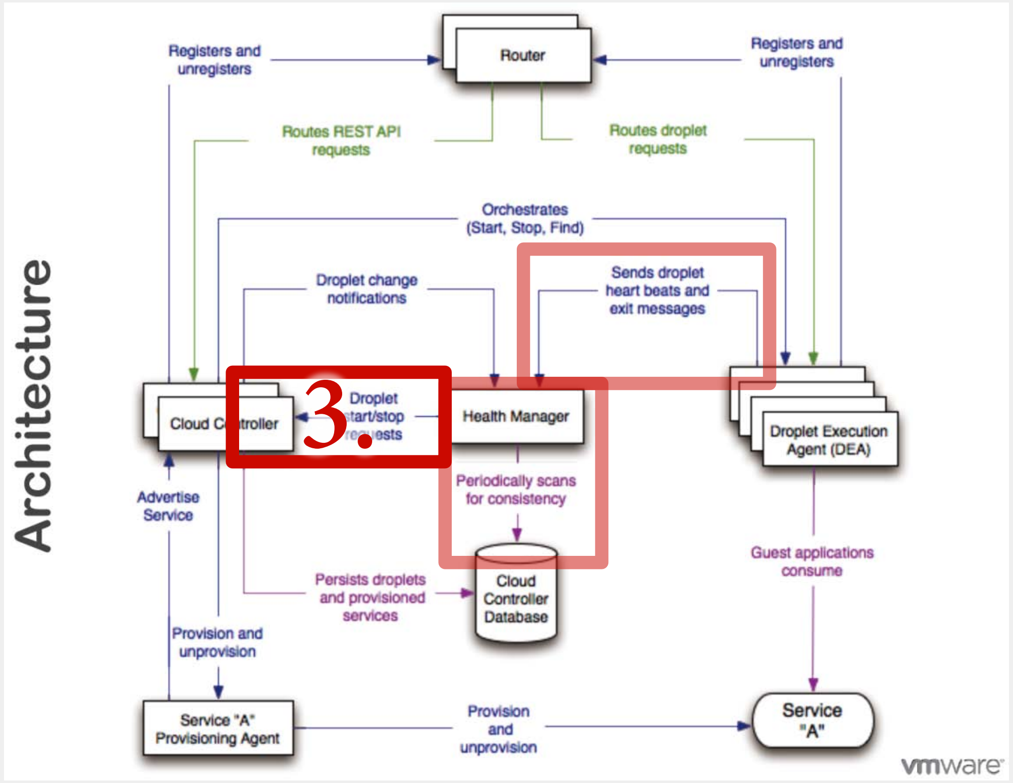
vcap/health_manager/lib/health_manager.rb

個々のアプリケーション分析

- 状態変更、必要な起動・終了

```
def analyze_app(app_id, droplet_entry, stats)
  now = Time.now.to_i
  update_timestamp = droplet_entry[:last_updated]
  quiescent = (now - update_timestamp) > @stable_state
  if APP_STABLE_STATES.include?(droplet_entry[:state]) && quiescent
    extra_instances = []
    missing_indices = []
    ……
    if missing_indices.any?
      start_instances(app_id, missing_indices)
    end
    if extra_instances.any?
      stop_instances(app_id, extra_instances)
    end
  end
end
```

vcap/health_manager/lib/health_manager.rb


Architecture
Router
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
Orchestrates (Start, Stop, Find)
Droplet change notifications
Sends droplet heart beats and exit messages
Cloud Controller
3.
Droplet start/stop requests
Health Manager
Droplet Execution Agent (DEA)
Advertise Service
Periodically scans for consistency
Guest applications consume
Persists droplets and provisioned services
Cloud Controller Database
Provision and unprovision
Service "A" Provisioning Agent
Provision and unprovision
Service "A"
vmware

@DEA

3. 異常を回復

インスタンスの終了

- NATSでインスタンスの終了を通知
  - CloudControllerが通知を受け取る
  - 実際の終了処理はCloudControllerから

```
def stop_instances(droplet_id, instances)
  droplet_entry = @droplets[droplet_id]
  last_updated = droplet_entry ? droplet_entry[:last_updated] : 0
  stop_message = {
    :droplet => droplet_id,
    :op => :STOP,
    :last_updated => last_updated,
    :instances => instances
  }.to_json
  NATS.publish('cloudcontrollers.hm.requests', stop_message)
  @logger.info("Requesting the stop of extra instances: #{stop_message}")
end
```

vcap/health_manager/lib/health_manager.rb

インスタンスの開始

- 設定によって2種類のフローが存在

```
def start_instances(droplet_id, indices)
  droplet_entry = @droplets[droplet_id]
  start_message = {
    :droplet => droplet_id,
    :op => :START,
    :last_updated => droplet_entry[:last_updated],
    :version => droplet_entry[:live_version],
    :indices => indices
  }
  if queue_requests?
    queue_request(start_message)
  else
    #old behavior: send the message immediately
    NATS.publish('cloudcontrollers.hm.requests', start_message.to_json)
    @logger.info("Requesting the start of extra instances: #{start_message}")
  end
end
```

vcap/health_manager/lib/health_manager.rb

インスタンス開始キュー

- 標準では20ms毎にインスタンス開始を通知

```ruby
def queue_requests?
  @request_queue_interval != 0
end
```

```ruby
def initialize(config)
  ……
  @request_queue_interval = config['intervals']['request_queue']
                                || 0.02
  ……
end
```

```ruby
def queue_request message
  #the priority is higher for older items, to de-prioritize flapping items
  priority = Time.now.to_i - message[:last_updated]
  priority = 0 if priority < 0 #avoid timezone drama
  key = message.clone
  key.delete :last_updated
  @logger.info("Queueing priority '#{priority}' request: #{message}, using key: #{key}. Queue size: #{@request_queue.size}")
  @request_queue.insert(message, priority, key)
end
```

vcap/health_manager/lib/health_manager.rb

インスタンス開始キューの消化

```
def run
  ……
    configure_timers
  ……
end
```

```
def configure_timers
  ……
  if queue_requests?
    EM.add_periodic_timer(@request_queue_interval) do
      unless @request_queue.empty?
        #TODO: if STOP requests are also queued, refactor this to be generic, particularly the log message
        start_message = encode_json(@request_queue.remove)
        NATS.publish('cloudcontrollers.hm.requests', start_message)
        @logger.info("Requesting the start of missing instances: #{start_message}")
        VCAP::Component.varz[:queue_length] = @request_queue.size
      end
    end
  end
end
```

vcap/health_manager/lib/health_manager.rb

ご清聴ありがとうございました

本スライドは下記URIで公開しています:

http://bit.ly/cfcrjp03-kuruma

http://kuruman.org/diary/2011/12/15/file/cloud-foundry-reading_kuruma.pdf