CPL: A Core Language for Cloud Computing

Oliver Bračevac
Sebastian Erdweg
Guido Salvaneschi
Mira Mezini
Deployment in the Cloud
Deployment in the Cloud
Deployment in the Cloud
Deployment in the Cloud
Deployment in the Cloud
Deployment in the Cloud

Specify **Deployment Patterns** with modular DSLs

```
  "Parameters": {
    "InstanceType": {
      "Description": "WebServer EC2 instance type",
      "Type": "String",
      "Default": "t2.small",
      "AllowedValues": [ "t2.micro", "t2.small",
                        "ConstraintDescription": "must be a valid EC2 instance type."
                    ],
    },
    "WebServer": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "InstanceType": { "Ref": "InstanceType" },  
        "UserData": { "Fn::Base64": { "Fn::Join": [ ""
          "#!/bin/bash -xe
          "yum update -y aws-cfn-bootstrap
          "/opt/aws/bin/cfn-init -v ",
          "--stack ", { "Ref": "AWS::StackName" },
          "--resource WebServer ",
          "--configsets wordpress_install ",
          "--region ", { "Ref": "AWS::Region" },"n"
        ] } },  //...
      },
      "Outputs": {
        "WebsiteURL": {
          "Value": { "Fn::Join": [ "", [ "http://", { "Fn::GetAtt": 
            [ "WebServer", "PublicDnsName" ], "wordpress" ] ] },
          "Description": "WordPress Website"
        }
      }
    }
  }
```
Deployment in the Cloud

Specify Deployment Patterns with modular DSLs

```json
{

  "Parameters": {
    "InstanceType": {
      "Description": "WebServer EC2 instance type",
      "Type": "String",
      "Default": "t2.small",
      "AllowedValues": [ "t2.micro", "t2.small" ],
      "ConstraintDescription": "must be a valid EC2 instance type."
    }
  }

  "Resources": {
    "WebServer": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "InstanceType": { "Ref": "InstanceType" },
        "UserData": { "Fn::Base64": { "Fn::Join": [ "", [ "#!/bin/bash -xe
          yum update -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-init -v --stack "", { "Ref": "AWS::StackName" }, "--resource WebServer",
          "--configsets wordpress_install",
          "--region", { "Ref": "AWS::Region" },"n"
        ] } } },
      "Outputs": {
        "WebsiteURL": { "Value": { "Fn::Join": [ "", [ "http://", { "Fn::GetAtt": [ "WebServer", "PublicDnsName" ] }, "/wordpress" ] ] }, "Description": "WordPress Website" }
      }
    }
  }
}
```
Deployment in the Cloud

Specify **Deployment Patterns** with modular DSLs

```json
{
  "Parameters": {
    "InstanceType": {
      "Description": "EC2 instance type",
      "Type": "String",
      "Default": "t2.small",
      "AllowedValues": ["t2.micro", "t2.small"],
      "ConstraintDescription": "must be a valid EC2 instance type."
    }
  },
  "Resources": {
    "WebServer": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "InstanceType": { "Ref": "InstanceType" },
        "UserData": { "Fn::Base64": { "Fn::Join": [
          "/bin/bash -xe
        , "yum update -y aws-cfn-bootstrap
        , "/opt/aws/bin/cfn-init -v "
        , "--stack", { "Ref": "AWS::StackName" }
        , "--resource WebServer",
        , "--confgsets wordpress_install",
        , "--region", { "Ref": "AWS::Region" }]
        ]}
      }
    }
  },
  "Outputs": {
    "WebsiteURL": {
      "Value": { "Fn::Join": [ "http://", { "Fn::GetAtt": [
        "WebServer", "PublicDnsName" ] } ] }
    }
  }
}
```
Deployment in the Cloud

Specify **Deployment Patterns** with modular DSLs

```yaml
create(      )

{ //...
  "Parameters": {
    "InstanceType": {
      "Description": "WebServer EC2 instance type",
      "Type": "String",
      "Default": "t2.small",
      "AllowedValues": [ "t2.micro", "t2.small" ],
      "ConstraintDescription": "must be a valid EC2 instance type."
    },
    ...
  },
  "Resources": {
    "WebServer": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "InstanceType": { "Ref" : "InstanceType" },
        "UserData": { "Fn::Base64": { "Fn::Join": [ "",
          "#!/bin/bash -xe
          yum update -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-init -v --stack "{{Ref::AWS::StackName}}" --resource WebServer --region "{{Ref::AWS::Region}}"
        ] }
      },
      ...
    },
    ...
  },
  "Outputs": {
    "WebsiteURL": {
      "Value": { "Fn::Join": [ "", [ "http://", { "Fn::GetAtt": [ "WebServer", "PublicDnsName" ] } ] ] },
      "Description": "WordPress Website"
    }
  }
}
```

```bash
#!/bin/bash -xe
yum update -y aws-cfn-bootstrap
/opt/aws/bin/cfn-init -v --stack "{{Ref::AWS::StackName}}" --resource WebServer --region "{{Ref::AWS::Region}}"
```
Deployment DSLs

Anatomy of Templates

Amazon CloudFormation
Deployment DSLs

Anatomy of Templates

- Parameters

```
"Parameters" : {
  "InstanceType" : {
    "Description" : "WebServer EC2 instance type",
    "Type" : "String",
    "Default" : "t2.small",
    "AllowedValues" : ["t2.micro", "t2.small"]
  }
}
```
Deployment DSLs

Anatomy of Templates

- Parameters
- Resource Specifications

```
"Parameters" : {
  "InstanceType" : {
    "Description" : "WebServer EC2 instance type",
    "Type" : "String",
    "Default" : "t2.small",
    "AllowedValues" : ["t2.micro", "t2.small"]
  } //...
},
"Resources" : {
  "WebServer" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : { ... },
  }
  "ElasticLoadBalancer" : {
    "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
    "Properties" : { ... }
  }
},
```

Amazon CloudFormation
Deployment DSLs

Anatomy of Templates

- Parameters
- Resource Specifications
- Output

```json
"Parameters": {
  "InstanceType": {
    "Description": "WebServer EC2 instance type",
    "Type": "String",
    "Default": "t2.small",
    "AllowedValues": ["t2.micro", "t2.small"]
  }
},

"Resources": {
  "WebServer": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
      ...
    }
  },
  "ElasticLoadBalancer": {
    "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
    "Properties": {
      ...
    }
  }
},

"Outputs": {
  "WebsiteURL": {
    "Value": {
      "Fn::Join": ["", [{"Fn::GetAtt": [
        "WebServer",
        "PublicDnsName"
      ], "/wordpress" }]]
    }
  }
}
```
Deployment DSLs

Unsafe Parameterization

"files" : {
  "/var/www/php/index.php" : {
    "content" : {"Fn::Join" : [ "" , [ "<html>
      "<head>
      "<title>Login</title>
      "</head>
      "<body>
      "<?php 
      {"Ref" : "SetupFunction"}, "; ?>
      "</body>
      "</html>
    ]]},
    "mode" : "000500",
    "owner" : "root",
    "group" : "root"
  }
}
Deployment DSLs

Two-phase Staging: **Indirect** topology changes at run time

Image taken from [Fernandez et al. ’13]
Deployment DSLs

**Inextensible Middleware Services**

```json
"ElasticLoadBalancer" : {
  "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
  "Properties" : {
    "Instances" : [ {"Ref":"Ec2Instance1"}, {"Ref":"Ec2Instance2"} ],
    "Listeners" : [ {
      "LoadBalancerPort" : "80",
      "InstancePort" : {"Ref":"WebServerPort"},
      "Protocol" : "HTTP"
    } ]
  }
}
```
The Cloud Platform Language

A core language for reasoning & studying

- Distributed cloud applications
- Programmable, extensible middleware
- Safe service composition
- Dynamic deployment

**Diagram:**

Application
Middleware
App + Middleware

Deploy || Run

Deploy → Run
First-class Servers

\[
\text{Fib} = \text{srv} \{ \\
\text{fib}(n, k) \triangleright \text{this} \# \text{acc}(n, 0, 1, k) \\
\text{acc}(n, a, b, k) \triangleright \\
\quad \text{if } n = 0 \\
\quad \text{then } k(a) \\
\quad \text{else } \text{this} \# \text{acc}(n - 1, b, a + b, k)
\}
\]

\[
\text{(spwn Fib)} \# \text{fib}(10, k) \rightarrow \text{i} \# \text{fib}(10, k) \rightarrow \text{i} \# \text{acc}(10, 0, 1, k) \rightarrow^* k(55)
\]
First-class Servers

\[
\text{Fib} = \text{srv} \{ \\
\quad \text{fib}(n, k) \triangleright \text{this} \# \text{acc}(n, 0, 1, k) \\
\quad \text{acc}(n, a, b, k) \triangleright \\
\quad \quad \text{if } n = 0 \\
\quad \quad \text{then } k(a) \\
\quad \quad \text{else } \text{this} \# \text{acc}(n - 1, b, a + b, k) \\
\}\]

Server Template

\[
(\text{spwn Fib}) \# \text{fib}(10, k) \rightarrow i \# \text{fib}(10, k) \rightarrow i \# \text{acc}(10, 0, 1, k) \rightarrow^* k(55)
\]
First-class Servers

Fib = srv {
  fib⟨n, k⟩ ▷ this#acc⟨n, 0, 1, k⟩
  acc⟨n, a, b, k⟩ ▷
    if n = 0
    then k⟨a⟩
    else this#acc⟨n - 1, b, a + b, k⟩
}

(spwn Fib)fib⟨10, k⟩ → i#fib⟨10, k⟩ → i#acc⟨10, 0, 1, k⟩ →* k⟨55⟩

Cloud spawns server instances from templates
Message Passing & CPS

\[
\text{Fib2} = \text{srv} \begin{cases} 
\text{fib}\langle n, k \rangle \\
\quad \text{if } n = 0 \lor n = 1 \\
\quad \quad k\langle n \rangle \\
\quad \text{else letk } x = \text{this}\#\text{fib}\langle n - 1 \rangle \\
\quad \quad \text{in letk } y = \text{this}\#\text{fib}\langle n - 2 \rangle \\
\quad \quad \quad \text{in } k\langle x + y \rangle 
\end{cases}
\]
Fib2 = srv { 
  fib\langle n, k \rangle \triangleright 
  \text{ if } n = 0 \lor n = 1 
  \quad \Downarrow \quad k\langle n \rangle 
  \text{ else let } x = \text{this}\#fib\langle n - 1 \rangle 
  \text{ in let } y = \text{this}\#fib\langle n - 2 \rangle 
  \text{ in } k\langle x + y \rangle 
} 

Direct style program
Fib2 = \texttt{srv} \{ 
\texttt{fib}\langle n, k \rangle \triangleright 
\text{if } n = 0 \lor n = 1 
\quad \text{let } k\langle n \rangle 
\text{else let } x = \texttt{this}\#\texttt{fib}\langle n - 1 \rangle 
\quad \text{in let } y = \texttt{this}\#\texttt{fib}\langle n - 2 \rangle 
\quad \text{in } k\langle x + y \rangle 
\}

\textbf{Desugaring:}
\texttt{letk } x = e_1\langle \bar{e} \rangle \text{ in } e_2 \quad \leadsto \quad e_1\langle \bar{e}, (\texttt{spwn (srv k\langle x \rangle \triangleright e_2))}\#k \rangle
Message Passing & CPS

Fib2 = srv {
  fib\langle n, k \rangle \triangleright
  if \ n = 0 \lor n = 1
    k\langle n \rangle
  else let k x = this\#fib\langle n - 1 \rangle
          in let k y = this\#fib\langle n - 2 \rangle
             in k\langle x + y \rangle
}

Desugaring:
letk x = e_1\langle \bar{e} \rangle in e_2 \leadsto e_1\langle \bar{e}, (spwn (srv k\langle x \rangle \triangleright e_2))\#k \rangle

Direct style program

Callback to server

There is only async message passing!
Fib2 = srv {
  fib\langle n, k \rangle \triangleright
    cache\#lookup\langle n, k, this\#run \rangle

  run\langle n, k \rangle \triangleright
    if n = 0 \lor n = 1
    then k\langle n \rangle
    else letk \ x = this\#fib\langle n - 1 \rangle
        in letk \ y = this\#fib\langle n - 2 \rangle
        in k\langle x + y \rangle
}
Message Passing & CPS

Service Composition

Fib2 = srv { 
  fib\langle n, k \rangle \triangleright 
  cache\triangleright lookup\langle n, k, this\rangle \triangleright 
  run\langle n, k \rangle \triangleright 
    if \ n = 0 \lor \ n = 1 
    then \ k\langle n \rangle 
    else letk \ x = this\triangleright fib\langle n - 1 \rangle 
     in letk \ y = this\triangleright fib\langle n - 2 \rangle 
     in \ k\langle x + y \rangle 
} 

Memo = srv { 
  lookup\langle n, k_{hit}, k_{miss} \rangle \triangleright 
    if hasEntry\langle n \rangle 
    then \ k_{hit}\langle get\langle n \rangle \rangle 
    else letk \ v = k_{miss}\langle n \rangle 
     in put\langle n, v \rangle; 
     k_{hit}\langle v \rangle 
    // details omitted ... 
}
Fib2 = \text{srv} \{ \\
\text{fib}\langle n, k \rangle \triangleright \\
\text{cache}\#\text{lookup}\langle n, k, \text{this}\#\text{run} \rangle \\
\text{run}\langle n, k \rangle \triangleright \\
\text{if } n = 0 \lor n = 1 \\
\text{then } k\langle n \rangle \\
\text{else let } k x = \text{this}\#\text{fib}\langle n - 1 \rangle \\
\text{in let } k y = \text{this}\#\text{fib}\langle n - 2 \rangle \\
\text{in } k\langle x + y \rangle \\
\} \\
\}

let cache = (spwn Memo) in (spwn Fib2)\#\text{fib}\langle 10, k \rangle
Message Passing & CPS

Service Composition

\[
\text{Fib2} = \text{srv} \{ \\
\text{\hspace{1em}fib}\langle n, k \rangle \triangleright \\
\hspace{2em} \text{cache\#lookup}\langle n, k, \text{this}\#\text{run} \rangle \\
\text{\hspace{1em}run}\langle n, k \rangle \triangleright \\
\hspace{2em} \text{if } n = 0 \lor n = 1 \\
\hspace{3em} \text{then } k\langle n \rangle \\
\hspace{2em} \text{else letk } x = \text{this\#fib}\langle n - 1 \rangle \\
\hspace{3em} \text{in letk } y = \text{this\#fib}\langle n - 2 \rangle \\
\hspace{4em} \text{in } k\langle x + y \rangle \\
\}
\]

let cache = (spwn Memo) in (spwn Fib2)#fib\langle 10, k \rangle

\[
\text{Memo} = \text{srv} \{ \\
\text{\hspace{1em}lookup}\langle n, k_{\text{hit}}, k_{\text{miss}} \rangle \triangleright \\
\hspace{2em} \text{if hasEntry}(n) \\
\hspace{3em} \text{then } k_{\text{hit}}\langle \text{get}(n) \rangle \\
\hspace{2em} \text{else letk } v = k_{\text{miss}}\langle n \rangle \\
\hspace{3em} \text{in put}(n, v); \\
\hspace{4em} k_{\text{hit}}\langle v \rangle \\
\hspace{2em} \text{// details omitted ...} \\
\}
\]
Message Passing & CPS

Service Composition

Fib2 = srv {
  fib⟨n, k⟩ ▷
    cache#lookup⟨n, k, this#run⟩

  run⟨n,k⟩ ▷
    if n = 0 ∨ n = 1
      then k⟨n⟩
    else letk x = this#fib⟨n − 1⟩
        in letk y = this#fib⟨n − 2⟩
        in k⟨x + y⟩
}

let cache = (spwn Memo) in (spwn Fib2)#fib⟨10,k⟩
let cache = (spwn Memo) in (spwn Fib2)#fib<10,k>
let cache = (spwn Memo) in (spwn Fib2)#fib\langle 10, k \rangle
Message Passing & CPS

Service Composition

Fib2 = srv {  
  fib\langle n, k \rangle \triangleright  
  cache\#lookup\langle n, k, this\#run \rangle  
  run\langle n, k \rangle \triangleright  
  if n = 0 ∨ n = 1  
  then k\langle n \rangle  
  else letk x = this\#fib\langle n - 1 \rangle  
      in letk y = this\#fib\langle n - 2 \rangle  
      in k\langle x + y \rangle  
}  

let cache = (spwn Memo) in (spwn Fib2)\#fib\langle 10, k \rangle
Message Passing & CPS

Service Composition

Fib2 = srv {
  fib⟨n, k⟩ ▷
  cache#lookup⟨n, k, this#run⟩
  run⟨n,k⟩ ▷
  if n = 0 ∨ n = 1 then k⟨n⟩
  else let k x = this#fib⟨n - 1⟩
    in let k y = this#fib⟨n - 2⟩
    in k⟨x + y⟩
}

let cache = (spwn Memo) in (spwn Fib2)#fib⟨10, k⟩
let cache = (spwn Memo) in (spwn Fib2)#{fib(10,k)}
let cache = (spwn Memo) in (spwn Fib2)#fib<10,k>
Concurrency & Join Patterns

[Fournet and Gonthier '96, Turon and Russo '11]

Fib3 = srv {
  fib⟨n, k⟩ ▷
   this#caller⟨k⟩ || this#invoke⟨n⟩

  invoke⟨n⟩ ▷
   if n = 0 ∨ n = 1
     this#res⟨n⟩
   else (spwn Fib3)#fib⟨n − 1, this#left⟩
     ||(spwn Fib3)#fib⟨n − 2, this#right⟩

  left⟨n⟩ & right⟨m⟩ ▷
   this#res⟨n + m⟩

  res⟨n⟩ & caller⟨k⟩ ▷ k⟨n⟩
}
Concurrency & Join Patterns

[Fournet and Gonthier ’96, Turon and Russo ‘11]

Fib3 = srv {
  fib⟨n, k⟩ ▷
  this#caller⟨k⟩ || this#invoke⟨n⟩

  invoke⟨n⟩ ▷
    if n = 0 ∨ n = 1
      this#res⟨n⟩
    else (spwn Fib3)#fib⟨n − 1, this#left⟩ || (spwn Fib3)#fib⟨n − 2, this#right⟩

  left⟨n⟩ & right⟨m⟩ ▷
    this#res⟨n + m⟩

  res⟨n⟩ & caller⟨k⟩ ▷ k⟨n⟩
}

Parallel execution
Concurrency & Join Patterns

[Fournet and Gonthier ’96, Turon and Russo ‘11]

Fib3 = srv {  
  fib\langle n, k\rangle ▷  
  \textbf{this}\#\text{caller}\langle k\rangle \ || \ \textbf{this}\#\text{invoke}\langle n\rangle  

  \text{invoke}\langle n\rangle ▷  
  \text{if } n = 0 \lor n = 1  
  \textbf{this}\#\text{res}\langle n\rangle  
  \text{else } (\text{spwn Fib3})\#\text{fib}\langle n - 1, \textbf{this}\#\text{left} \rangle  
  \ || (\text{spwn Fib3})\#\text{fib}\langle n - 2, \textbf{this}\#\text{right} \rangle  

  \text{left}\langle n\rangle \ & \ \text{right}\langle m\rangle ▷  
  \textbf{this}\#\text{res}\langle n + m\rangle  

  \text{res}\langle n\rangle \ & \ \text{caller}\langle k\rangle ▷ k\langle n\rangle  
}
Server Management

Migratable = srv {
    migrate<dest> ▷
        let img = snap this
        in (dest<img>
            || repl this 0)

    // ...
}

Migratable = srv {
    migrate(dest) ▷
    let \( \text{img} = \text{snap this} \)
    in (dest(img)
        || repl this 0)

    // ...
}

Retrieve server image from cloud
Server Management

Migratable = srv {
  migrate(dest) ▷
  let \textcolor{red}{\textit{img} = \textit{snap this}}
  in (dest|\textcolor{blue}{\textit{img}})
    || \textcolor{blue}{\textit{repl this 0}}

  // ...
}

Retrieve server image from cloud

Replace with different image
(inactive server in this case)
Server Placement

Are First-Class Servers VMs?

\[
\text{Fib2} = \text{srv } \{ \\
\text{fib}\langle n, k \rangle \triangleright \\
\text{if } n = 0 \lor n = 1 \\
\quad k\langle n \rangle \\
\text{else letk } x = \text{this#fib}\langle n - 1 \rangle \\
\quad \text{in letk } y = \text{this#fib}\langle n - 2 \rangle \\
\quad \text{in } k\langle x + y \rangle \\
\}
\]
Fib2 = srv { 
  fib⟨n, k⟩ ▷
  if n = 0 ∨ n = 1
    k⟨n⟩
  else letk x = this#fib⟨n − 1⟩
    in letk y = this#fib⟨n − 2⟩
    in k⟨x + y⟩
}
Server Placement

Are First-Class Servers VMs?

Fib2 = \texttt{srv} \{
  \texttt{fib}⟨n, k⟩ \triangleright \\
  \textbf{if} n = 0 \lor n = 1 \\
  \quad k⟨n⟩ \\
  \textbf{else} \texttt{letk} x = \texttt{this} \#\texttt{fib}⟨n − 1⟩ \\
  \textbf{in} \texttt{letk} y = \texttt{this} \#\texttt{fib}⟨n − 2⟩ \\
  \textbf{in} k⟨x + y⟩
\}
Server Placement
Are First-Class Servers VMs?

Fib2 = srv
  \{ fib\langle n, k \rangle \triangleright
      if \ n = 0 \lor n = 1
          k\langle n \rangle
      else letk x = this#fib\langle n - 1 \rangle
                 in letk y = this#fib\langle n - 2 \rangle
                    in k\langle x + y \rangle
  \}

Current implementation: Annotations for locality
Future work: Automated placement by run time
Server & Service Combinators
Server & Service Combinators

Load Balancing

work<...>
Server & Service Combinators

Load Balancing
Server & Service Combinators

Load Balancing
Server & Service Combinators

Load Balancing
Server & Service Combinators

Load Balancing

work<w,k>

work<...>
load?<...>

work<...>
load?<...>

work<...>
load?<...>

work<...>
load?<...>

next?<...>
Server & Service Combinators

Load Balancing

work<...>
load?<...>
work<...>
load?<...>
work<...>
load?<...>

next?<...>

work<w,k>
Server & Service Combinators

Load Balancing
Server & Service Combinators

Load Balancing
Server & Service Combinators

Load Balancing
Server & Service Combinators

Load Balancing

- $k<\text{res}>$
- $\text{work}<w,k>$
- $\text{work}<\ldots>$
- $\text{load}?<\ldots>$
- $\text{next}?<\ldots>$
- $1$
- $\text{load}?$
- $\text{work}<\ldots>$
- $\text{load}?<\ldots>$
- $\text{work}<\ldots>$
- $\text{load}?<\ldots>$
Server & Service Combinators

work\langle...\rangle

k\langle res \rangle

work\langle w, k \rangle
Server & Service Combinators

First class deployment pattern!

```
{
  "Parameters": {
    "InstanceType": {
      "Description": "WebServer EC2 instance type",
      "Type": "String",
      "Default": "t2.small",
      "AllowedValues": ["t2.micro", "t2.small"],
      "ConstraintDescription": "must be a valid EC2 instance type."
    }
  },
  "Resources": {
    "WebServer": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "InstanceType": { "Ref": "InstanceType" }
      },
      "UserData": {
        "Fn::Base64": {
          "Fn::Join": [
            "",
            ["#!/bin/bash -xe
             yum update -y aws-cfn-bootstrap
             /opt/aws/bin/cfn-init -v 
             --stack "", { "Ref": "AWS::StackName" } 
             --resource WebServer 
             --region "", { "Ref": "AWS::Region" } 
             --configure-parameters, install 
             --region "", { "Ref": "AWS::Region" } 
          ]
        }
      }
    }
  },
  "Outputs": {
    "WebsiteURL": {
      "Value": {
        "Fn::Join": [
          "",
          ["http://", { "Fn::GetAtt": [ "WebServer", "PublicDnsName" ] }, "/wordpress"
        ]
      },
      "Description": "WordPress Website"
    }
  }
}
```
Server & Service Combinators

First class deployment pattern!

Further combinators in the paper:
MapReduce
Failure Recovery
Structural Type System
Based on System Fₜₜₜ

Depth- & Width-Subtyping

Fib2 :: srv { fib: \langle\text{Int}, \langle\text{Int}\rangle\rangle } 
Fib :: srv { fib: \langle\text{Int}, \langle\text{Int}\rangle\rangle, acc: \langle\text{Int, Int, Int, \langle\text{Int}\rangle}\rangle } 
Fib \triangleleft\triangleright Fib2

Bounded Polymorphism

MKBalanced :: \forall \alpha \forall \omega \triangleleft\triangleright \text{TLAWorker}[\alpha]. \text{srv} \{ \text{make}: (\text{List}[\omega], \text{Choose}[\omega]) \rightarrow \text{TWorker}[\alpha] \}

Service Requests “Can’t Go Wrong”: Soundness Proof in Techreport
Summary & Outlook

Cloud Deployment DSLs
- Inextensible middleware
- Two-phase staging
- Unsafe parameterization

Paper
- Formal syntax & semantics
- Combinators for load balancing, fault tolerance, MapReduce
- Concurrent, single machine implementation in Scala

CPL
- Middleware combinators
- Dynamic Deployment
- First-class servers, CPS, Types

Ongoing Work
- Distributed, cloud-based implementation
\[ \lambda x. \ e \leadsto \text{spwn} \ (\text{srv} \ \text{app} \langle x, k \rangle \triangleright T(e, k)), \text{ where } k \text{ is fresh} \]

\[ T(\lambda x. \ e, k) = k\langle \text{spwn} \ (\text{srv} \ \text{app} \langle x, k \rangle \triangleright T(e, k)) \rangle \]

\[ T((f \ e), k) = T(f, (\text{spwn} \ (\text{srv} \ k_1\langle v_f \rangle \triangleright \text{where } v_f \text{ is fresh}) \]

\[ T(e, (\text{spwn} \ (\text{srv} \ k_2\langle v_e \rangle \triangleright \text{where } v_e \text{ is fresh}) \]

\[ v_f \# \text{app} \langle v_e, k \rangle \rangle \rangle \rangle \# \langle k_2 \rangle \rangle \rangle \# k_1 \]

\[ T(e, k) = k\langle e \rangle \]

[Fournet and Gonthier ’96]
Derived Syntax

\[
\text{let } x = e_1 \text{ in } e_2 \leadsto (\text{spwn} (\text{srv} \ \text{let} \langle x \rangle \triangleright e_2))\#\text{let} \langle e_1 \rangle
\]

\[
\text{letk } x = e_1\langle \bar{e} \rangle \text{ in } e_2 \leadsto e_1\langle \bar{e} \rangle, (\text{spwn} (\text{srv} k\langle x \rangle \triangleright e_2))\#k
\]

\[
\text{thunk } e \leadsto \text{srv force} \langle k \rangle \triangleright k\langle e \rangle
\]