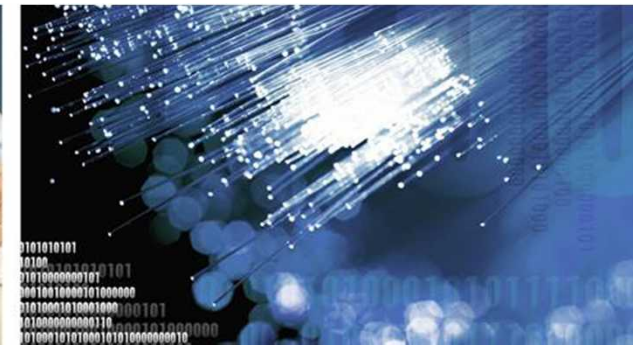


*Exceptional service in the national interest*



# Cloud Development Strategies

A 2015 NLIT “Fueling the Fire in IT” presentation

John G Miner, Mgr of Enterprise Application Architecture and Cloud Strategies

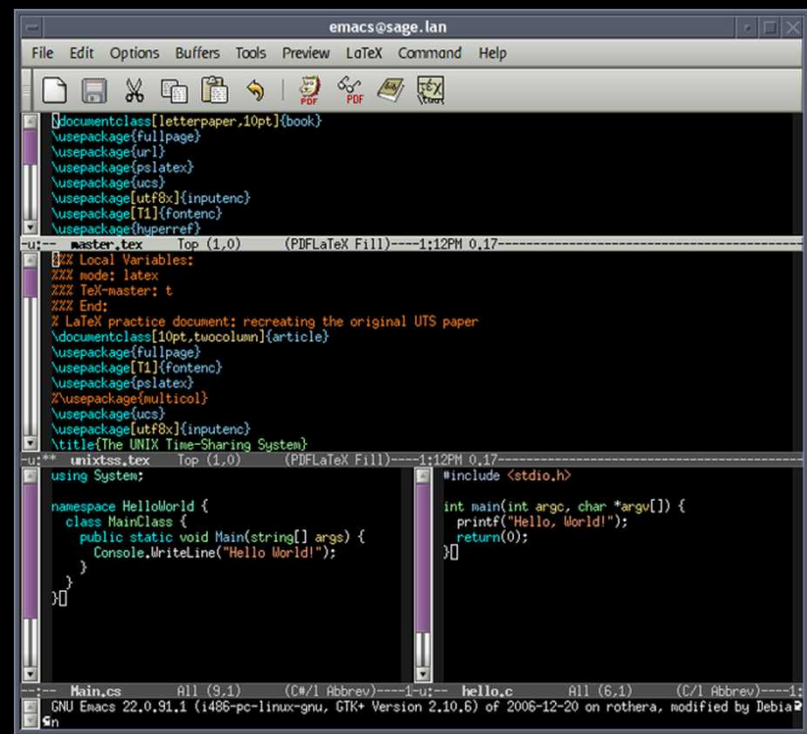


Long ago in a galaxy far away  
We memorized syntax,  
Fought compilers  
Built monolithic apps

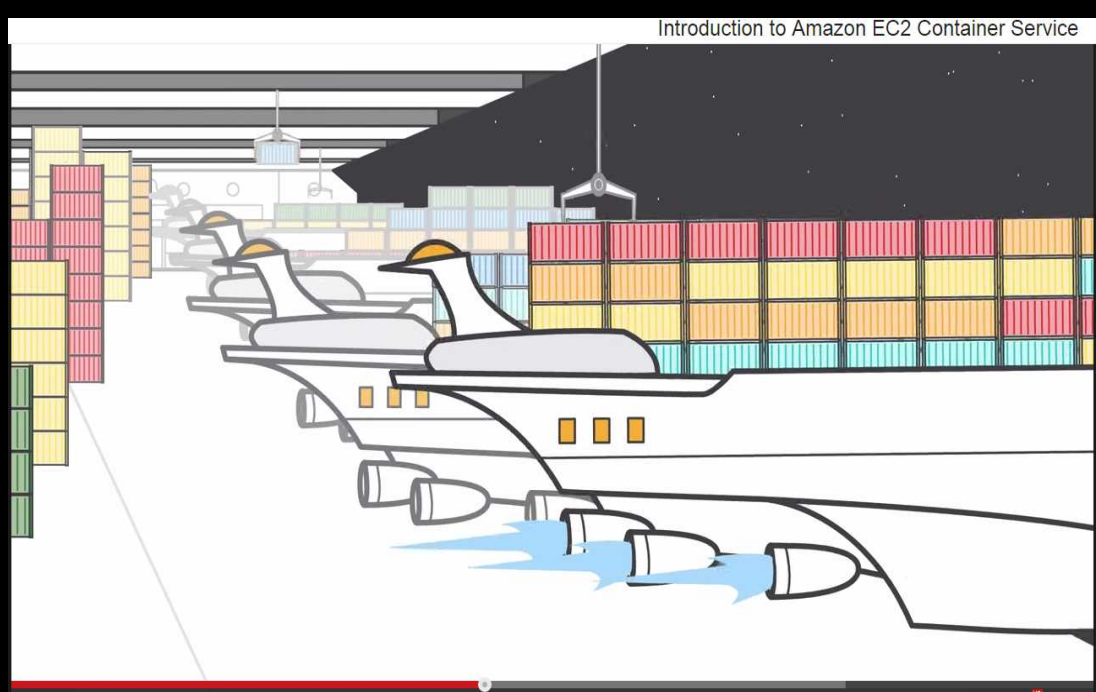
But Now we must  
Loosely couple services  
Containerize configurations  
Continuously Release

How can Lab IT groups  
Adapt fast enough?

Key is understanding that  
Commercial Clouds provide  
Developer experimentation  
Space ...



```
emac@sage:lan
File Edit Options Buffers Tools Preview LaTeX Command Help
[Icons]
\documentclass[letterpaper,10pt]{book}
\usepackage{fullpage}
\usepackage{url}
\usepackage{pslatex}
\usepackage{ucs}
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{hyperref}
-- master.tex Top (1,0) (PDFLaTeX Fill)----1:12PM 0.17--
[ZZ Local Variables:
XXX mode: latex
XXX TeX-master: t
XXX End:
% LaTeX practice document: recreating the original UTS paper
\documentclass[10pt,twocolumn]{article}
\usepackage{fullpage}
\usepackage[T1]{fontenc}
\usepackage{pslatex}
\usepackage{multicol}
\usepackage{ucs}
\usepackage[utf8x]{inputenc}
\title{The UNIX Time-Sharing System}
-- unixts.tex Top (1,0) (PDFLaTeX Fill)----1:12PM 0.17--
using System;
#include <stdio.h>
namespace HelloWorld {
class MainClass {
public static void Main(string[] args) {
Console.WriteLine("Hello World!");
}
}
}
int main(int argc, char *argv[]) {
printf("Hello, World!");
return(0);
}
-- Main.cs All (9,1) (C#/1 Abbrev)----1:12PM 0.17-- hello.c All (6,1) (C/1 Abbrev)----1:12PM 0.17--
GNU Emacs 22.0.91.1 (1486-pc-linux-gnu, GTK+ Version 2.10.6) of 2006-12-20 on rothera, modified by Debian
```



# Cloud equivalent of embryological parallelism

- How IT people tend to adopt cloud:
  - Build an app with an installer and build it on full cloud server and release to prod
  - Deploy a canned app, tailor and release to prod
  - Deploy Package ( JAR/WAR) to PaaS
  - Deploy Service in a Container
  - Deploy a Lambda from a embedded editor
- What it would look like in reverse
  - Have external events be a proxy for your event triggers
  - Create Lambda function respond to 'events'
  - Buy a containerized service that outperforms the lamda and import it
  - Collaborate with contractors on a java app, package for internal PaaS to use the service
  - Run a installer script from a CDN that starts solution on laaS assuming service exists
  - Create a new software concept on local host from best in bred snippets that removes the need for the service.

- Zapier > AWS SNS > Lambda Demo
- <http://aws.amazon.com/lambda/>



The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, 'Services', and 'Edit' dropdown. The user's name 'John G Miner' and location 'N. Virginia' are visible. The main heading is 'Lambda: Edit/Test myfirstlamda'. Below this, there are two tabs: 'Function code' (selected) and 'Upload a ZIP file'. The code editor displays the following JavaScript code:

```
1 console.log('Loading event');
2
3 exports.handler = function(event, context) {
4   console.log('value1 = ' + event.key1);
5   console.log('value2 = ' + event.key2);
6   console.log('value3 = ' + event.key3);
7   context.done(null, 'Hello World'); // SUCCESS with message
8 };
```

Below the code editor, there are two expandable sections: 'Change function configuration and role' and 'Change advanced settings'. The 'Change advanced settings' section is expanded, showing a note: 'Changing memory or timeout will impact your cost. See the pricing guide for details.' At the bottom right, there are two buttons: 'Go to function list' and 'Invoke'. At the very bottom, there's a 'Memory (MB)' dropdown set to '128'.

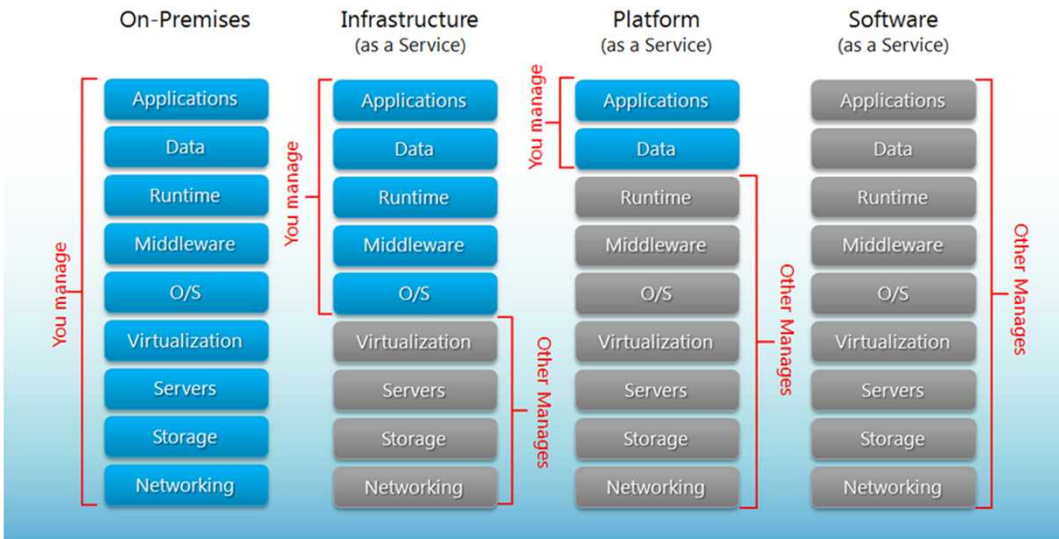
# Software Source Progression

- Developed In place
- Developed Remotely
- Commercial Off the Shelf COTS ( The shelf has disappeared )
- Gov Off the Shelf GOTS (Where was the shelf ? )
- Commercial from the Cloud
- Developed on the Cloud
- Developed For the Cloud (only)
- Software as a Service
- Commercial Software as a Service
- Gov Software as a Service (?)

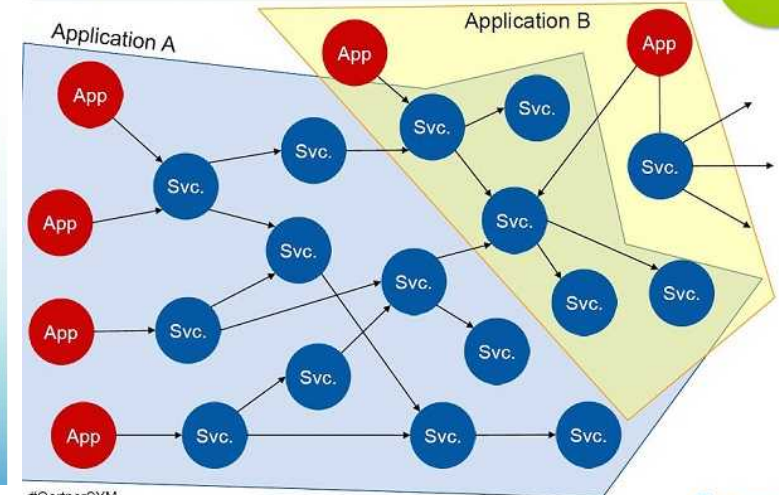
# Enterprise App Cloud Strategies

- Enterprise Application evolution from two key strategies:
  - Construct Solutions from Atomic Services
  - Deploy to Platform as a Service wherever possible
- To enable faster innovation, we have two key principles
  - Develop in the best development environment for the *project* team
  - Deploy to the best environment for the enterprise

## Separation of Responsibilities



## The New Model: Apps and Services Architecture



#GartnerSYM

© 2014 Gartner, Inc. and/or its affiliates. All rights reserved.

Gartner

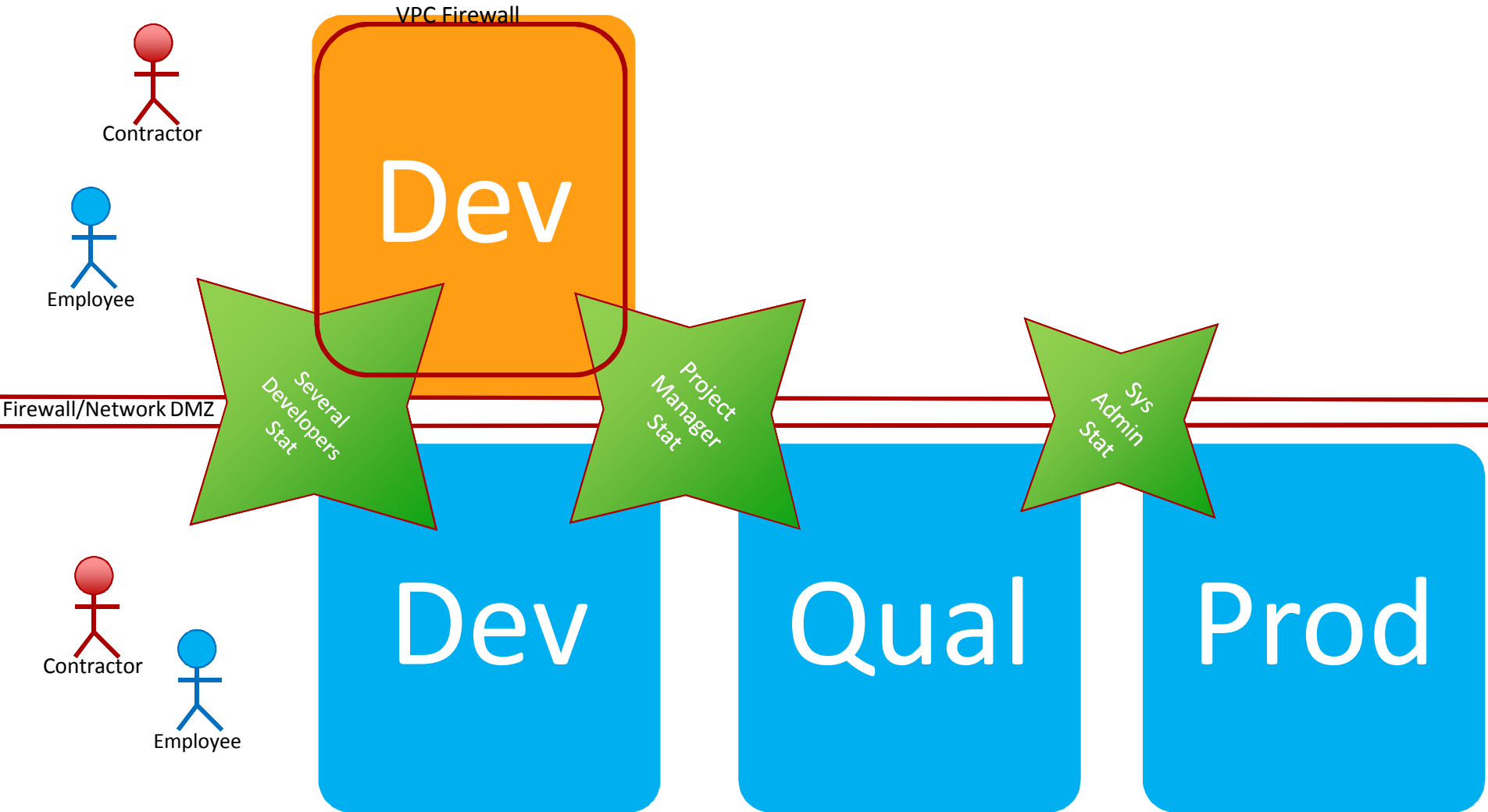
# Software Defined Everything

- Router network
- Security model
- Server, cluster, installer
- Use inside->out port 80 SSL REST service for communications as much as possible
- Use Platform as a Service for portability
- Use Database as a Service for ease and obscurity
- The Developer can now change the 'environment' with tools they know

```
"myVPC": {
  "Type": "AWS::EC2::VPC",
  "Properties": {
    "CidrBlock": {"Ref": "myVPCCIDRRange"},
    "EnableDnsSupport": false,
    "EnableDnsHostnames": false,
    "InstanceTenancy": "default" }
},
"myInstance" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "ImageId": {
      "Fn::FindInMap": ["AWSRegionToAMI", {"Ref": "AWS::Region"}, "64"] },
    "SecurityGroupIds" : [{"Fn::GetAtt": ["myVPC", "DefaultSecurityGroup"]}],
    "SubnetId" : {"Ref" : "mySubnet"}
  }
}
```

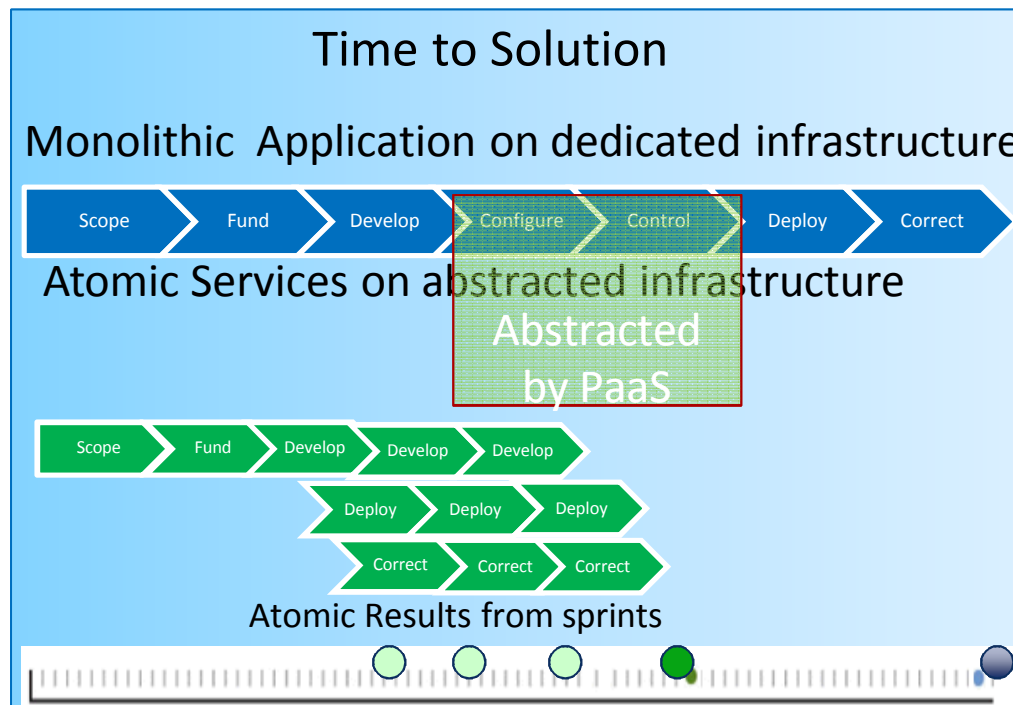
JSON file example snippet from Amazon,  
Dynamically configurable by code

# External Development Colaboration

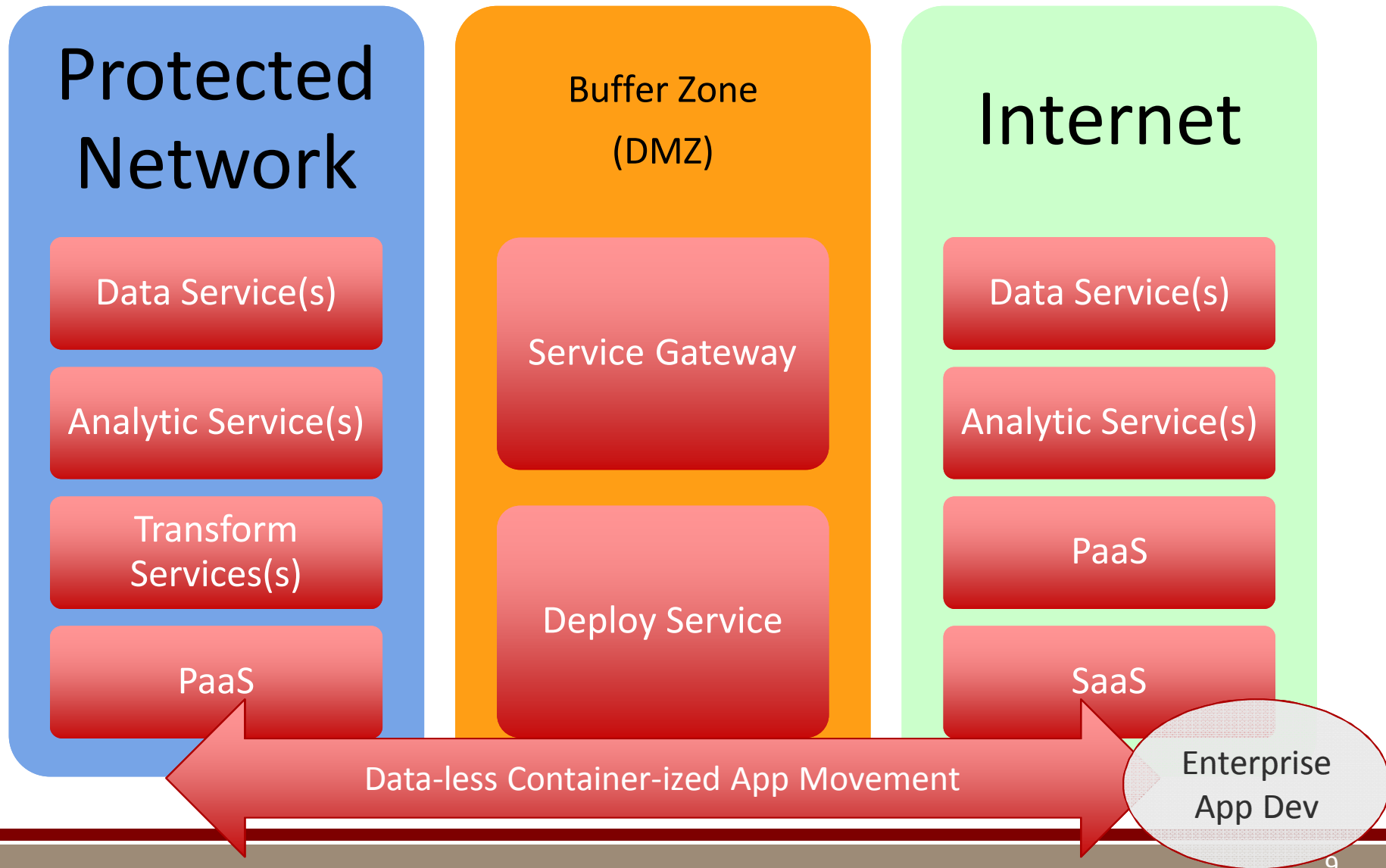


# The Goals from using these Strategies:

- Increase the productivity of the Ent App Developer
- Reduce Time-to-Solution (TtS) , Atomic services used by other needs
- Reduce the switching cost for moving solutions based on business need
- Provide industry capabilities to developers:
  - encryption at rest & transport, high availability 24x7, and programmable elasticity



# Systematic Vision



# Example Opportunity

- Human Resource Management
  - Create atomic data sources with CRUD services: Who, What, When
  - Uses diverse analytics
- Wolfram Alpha like (similar to PeopleSoft Weblets Arch)
  - Development and acquisition of of atomic analytics
    - Each capable of being cast on different data sets
  - Acquisition and vetting of datasets
    - Each owned by independent data Stewarts from independent sources
- In a cloud enabled enterprise networked
  - All stakeholder groups could contribute atomic analytics including vendors
  - All Data Stewarts can supply the independent appropriately authenticating service of record.

# Three Testing/PoC Methods, of Many Possible

- App Streaming
  - <http://aws.amazon.com/appstream/>
  - Build remotely and test/demo just screen bits
- Code synchronizing
  - <http://aws.amazon.com/s3/>
  - S3 to Cloudfront CDN
  - Git Hosting ...
- Service brokering ( api gateway model )
  - [PeopleSoft RESTful APIs](#)
  - Create restful services that can be abstracted by internal gateways that have security and usage polices
  - Allows for proof points that can be moved inside and start working after a single redirect
- All of these support the principle of getting the project team the best environment to innovate in.