Infrastructure Manager

By

Jeremy Ridge and Ryan Fields

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Jeremy Ridge — Date: 4/17/17

Ryan Fields — Date: 4/17/2017

Russell McMahon, Faculty Advisor — Date: 4/17/2017

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2017

Table of Contents

Table and Figures

Abstract

Administrating a large computer infrastructure is difficult when questions or issues arise. Infrastructure Manager will provide all computerized system infrastructure information in one web based application. Often the administrator is required to log into several applications, or run several reports to answer questions or troubleshoot issues. The creation of one centralized web based application that displays all of the information needed for infrastructure support personnel saves administrative time, and reduces costly downtime to the end user. The primary client for this application is University of Cincinnati, but could also be developed for commercial use for almost any infrastructure. With the consolidation and customization of already existing reporting/management information into one application, Infrastructure Manager will save time and money.

Introduction

The University of Cincinnati is one of the largest employers in the Cincinnati area, and also has an annual enrollment of 40,000 students which makes it the second largest university in Ohio as well as in the top 50 for number of students in the United States. The computing environment needed to support the primary ERP (Enterprise Resource Planning) systems that the university uses to maintain both the human resources and financial business, as well as the student body information system has become very vast. The university uses SAP as its ERP system for both Human Resources and Financial business. This is a large virtual environment housed in one rack within the UC datacenter. The university also uses PeopleSoft as its ERP for its Student Information System, also in a virtual environment that is housed in one rack alongside the SAP ERP. The university has recently decided to merge the two infrastructure systems into the same environment, making management easier, from a people perspective. But the information for each system is very difficult to gather without using multiple resources to answer some of the most basic questions. The application we are developing will provide a single pane of glass to view information for all of the virtual systems that are in use for the two most important environments that are in use at the university.

Project Description

We are developing and implementing a central web based interface for monitoring many of the applications and hardware in the university BCS (Business Core Systems) infrastructure and SIS (Student Information System). Primarily this is for the Student Information System ERP system (Catalyst) vm environment, and for the Business Core Systems vm environment, which includes SAP (UCFlex). This application will be very beneficial for the system administrators and management as it will give them one location for accessing data in regards to the systems they are managing. Instead of logging into separate systems or applications to retrieve the information they are looking for they can use one application. This will also help with management decisions in regards to upgrades and resource purchasing, as well as help with eliminating downtime due to resource constraints or failures

Design Objectives

Our application is web based and will be able to be accessed using any browser. Also the information that will be available to the users will be read only. As an added measure of security we are researching a logon authentication to utilize LDAP from UC that is currently in use with most applications. All of the systems that are being reported on exist on UC's back end network which is not generally available to those already on the UC general public network. To make this available to those on UC's general public network we will be making a DNS entry for the application to be registered on the network. This will also be limited by IP subnet when this is configured in the F5 firewall as an added security feature. Upon login the user will have the

option to choose the System Administrator data or the Database Administrator data. Once the user is in the application they will then be able to look at multiple e systems and find data regarding location, cpu usage, memory usage/allocation, version, and disk drive information.

Security

This application will not be accessible on the internet, and will only be available to users that are on UC's network. This provides a layer of security, but the policy for UC is BYOD (Bring Your Own Device), so this application cannot be left unsecure. As an added level of security the information viewed in the application will be read only for all users, as it is primarily informational data. To continue with the security, we are working to add logon authentication against the UC active directory that may then be limited to a certain IP range of users to make the information available only to those who need it. Lastly, a CORS security module was implemented to allow cross site scripting between the JSON server and the web server communication. This was necessary since this is blocked by default but is able to be limited with an access list.

Methodology

The design approach utilized in this project was a straight design and release. This is a different approach since the primary developers are the primary users as well. Since this is also an "internal" application to only be utilized by the university this allowed for us to make the decisions as to what is important and should be featured. The environment is entirely comprised of virtual machines which allows for restores after changes with relative ease, as long as backups or snapshots are maintained prior to the change. Also since the application is entirely read only we do not see the need for changes going forward as once it is configured it will be operational. If the situation arises where there needs to be a major change in regards to OS or some form of a code change the entire system, all node.JS and database servers, can be cloned and turned into a test environment to configure and test new changes.

User Profile

The target user profile for Infrastructure Manager is primarily system administrators and management. This group is technical and understands their environment and knows where and what to look at when an issue arises. While this is the primary target user profile, this application can be utilized by many others within an IT infrastructure. With a read only interface this application can be utilized by many other support personnel such as application administrators, database administrators, and developers. The application will provide one location to look at many different aspects of their system infrastructure. This includes versioning of systems OS/DB, resource consumption of cpu/memory, and location.

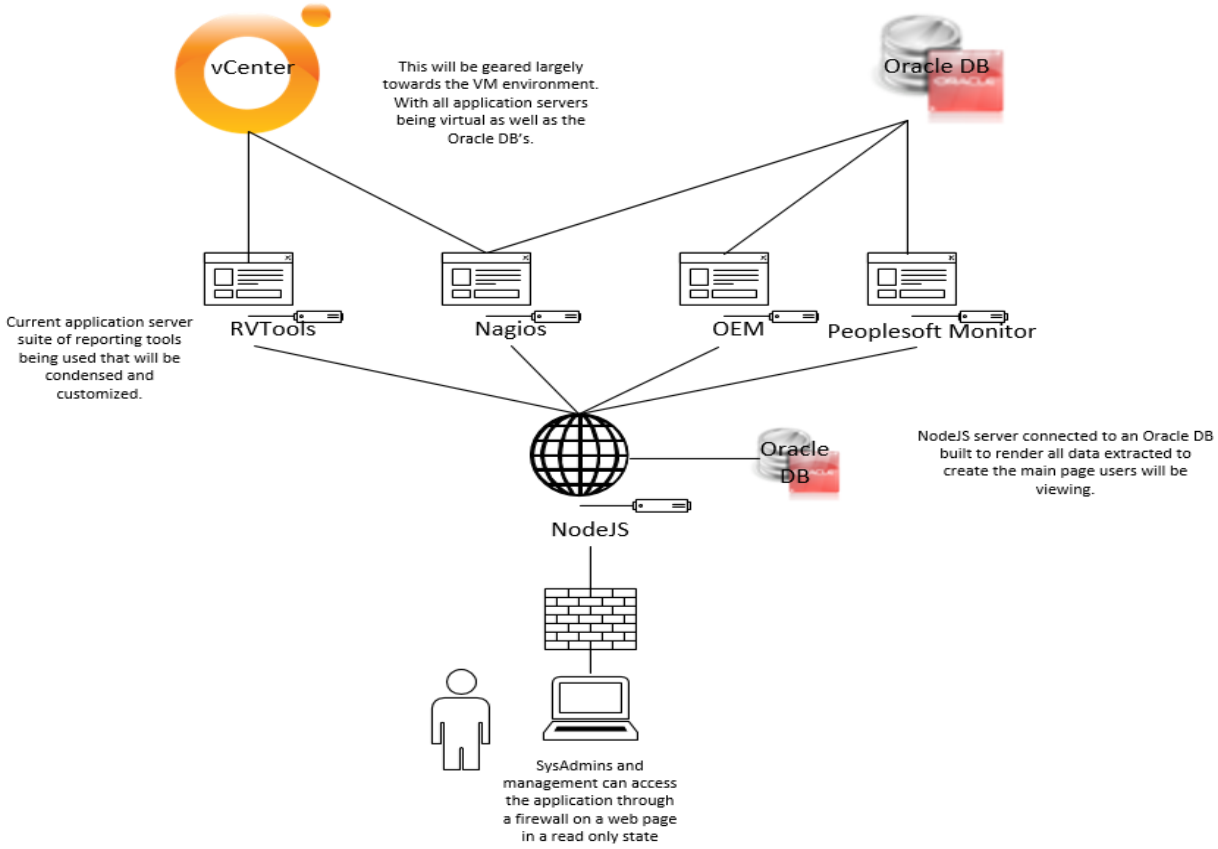

Figure 1: User Profile

Use Case Diagram

Infrastructure Manager

Here is a use case diagram displaying the different functions an administrator or manager can accomplish with our application.

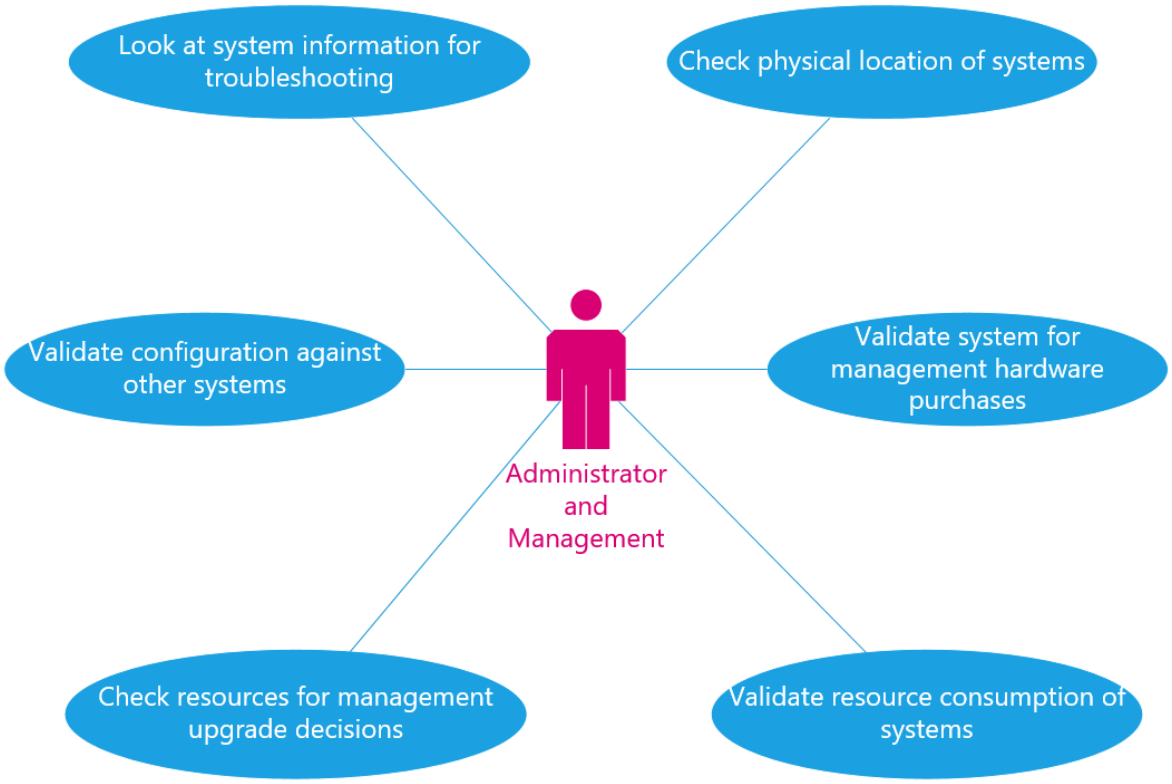

Figure 2: Use Case Diagram

Proposed Budget

The costs represented below are real world prices. We are utilizing the university software, hardware and networking so there is no cost to the college, but what it would cost in the private sector. This application will be utilized within the university using all already purchased hardware and software. The project is using a purchased version of Oracle database but there is a free version in Oracle XE. The budget below is proposed using free software in the form of Oracle Linux OS and Oracle XE database software. The F5 cost below would be the cost for a company to have an employee, or possibly an outside contractor, to configure their F5 device for the application to be made available on the network. A private sector company may have a different device for their firewall or network configuration.

No.	Item	No. Unit	Cost/Unit	Total
	Networking			
1	F5 Config	40	$35.00	$1,400.00
	Software Development			
2	Labor	120	$85.00	$10,200
	Software			
3	Oracle XE DB	1	0	0
4	Oracle Linux	1	0	0
				$11,600

Table 1: Proposed Budget

Testing

Testing Timeline Estimate

Fall Semester	Week 1	• Analyze design, decide what features to include • Determine how to get data from all sources
	Week 2	• Discuss tools to use • Determine what data is important • Discus features
	Week 3	• Research connectivity • Design and limitations • Test data extraction
	Week 4	• Build database • Build initial nodeJS for DB interface
	Week 5	• Build node.JS for web service • Build and test proof of concept
	Week 6	• Complete proof of concept • Prepare for presentations
	Week 7 - 10	• Prepare for presentation • Present when needed

Spring Semester	Weeks 1 - 3	• Analyze project at this point • Discuss features to add
	Weeks 4 to 2 weeks pre-expo	• Improve application • Finish added features
	2 Weeks pre-expo - Expo	• Polish final build • Prepare for expo

Estimated Gantt Chart

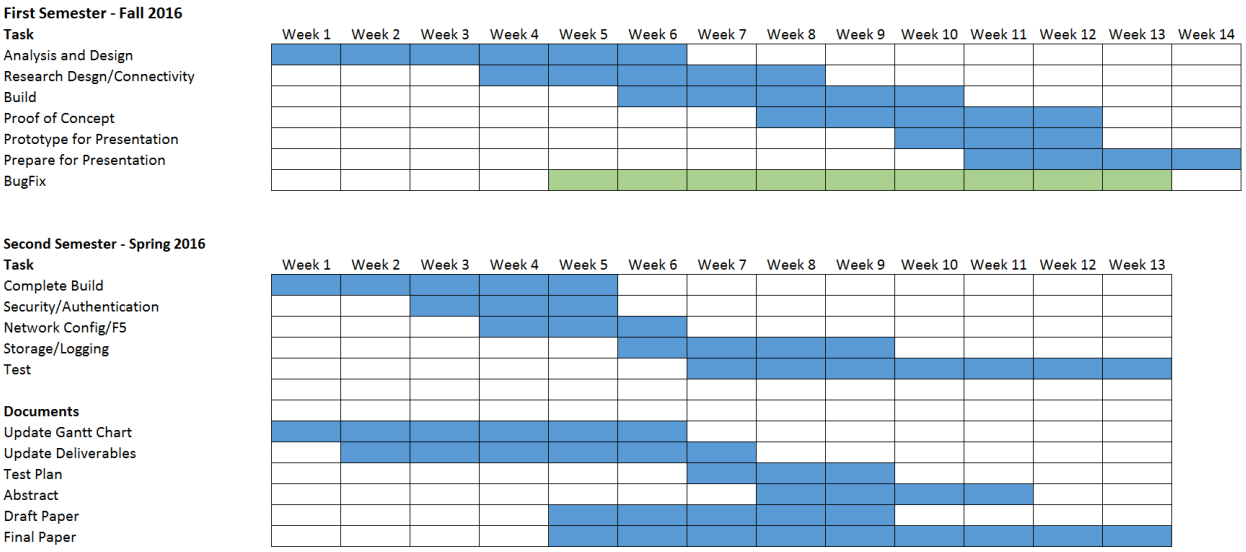

Table 2: Gantt Chart

Problems Encountered

The issues encountered in this project were minimal but we did have some. Due to the fact that this team has the resources readily available to us to work through them, the issues were few and far between. The biggest issues the team encountered were compatibility, running services and data extraction from vCenter.

- Issue 1: nodeDB driver compatibility with Oracle Enterprise Linux 6

  Initially we had planned on using an existing machine that runs systems management utilities already. This system was running Oracle Enterprise Linux 6 (OEL6). Some of the node modules, and most importantly, the Oracle nodeDB driver would not function properly on OEL6. For full functionality the nodeDB driver required OEL7. The machine that we had already began using could not be taken down for an upgrade as

there was not sufficient time to test the OS upgrade and validate other tools would not be impacted. The decision was made to create a separate middleware system running OEL7 to handle the nodeJS applications. Since we have plenty of virtual machine resources this did not cause a major issue other than taking up time to troubleshoot and eventually reinstall.

- Issue 2: Getting the nodeJS apps to run as a system service

Currently, the only way our node apps run, are with a logged in user session running the applications. We are working on getting these to run as a system service that doesn't require a logged in session. This is not a large issue currently and we plan on resolving this by meeting with the app team for the Catalyst project to find out how they are running their nodeJS web services.

- Issue 3: Extracting data and placing it an NFS mount

For extracting data from vCenter server we used RVTools. This tool extracts a lot of data from vCenter in .csv or .xls format. While the tool functioned normally it was running on a windows server. To get this data from windows server to an NFS mount that our application can read, proved to be difficult. While windows servers will not maintain NFS mounts, it was decided to use WinSCP so the data extraction could run as a scheduled job and the WinSCP utility could run and place the data in an NFS mount

Test Plan

Overview

This section will discuss the testing steps and methodology used for the Infrastructure Manager application, which is to be used as a guide. The following individuals should use this section:

- Administrators
- Managers
- Database Administrators
- Developers

Objective

The objective of testing is to verify the application itself works as a whole, and also to ensure all of the deliverables that we wanted to implement are working as expected. Since this is an internal application, created solely for UC employees, the testing will be restricted to that group. The tests are designed to test the features we have implemented within the application. Since the application will be used by our group within UC as well as us, we are implementing what we feel is the most useful features.

Deliverables for Infrastructure Manager

1. Application access only on UC network.
2. Logon option of SysAdmin or DBA from the logon screen.
3. Data available is “read only”.
4. Search functionality for a particular system.

5. System Data that is most useful is displayed for the user. System information updated at least daily if not more frequent.

Logging and test reporting

Errors and issues that are found during testing will be documented among the team. This will then be determined and resolved. As the application is not extremely complex we are testing as we are completing the build, so issues may be minimal.

System Testing

Infrastructure Manager will be tested as a complete application with everything tested as one unit. This is due to the automation and normalization of data being imported and the features we implemented with data being displayed in a "read only" format. This will help to ensure we do not have any issues, and that we will expect to see the performance we intend.

Testing Procedure

The following are steps that are needed for testing:

- Create a test scenario for both DBA and SysAdmin login ID's
- Prepare documentation to produce and record test results
- Specify issues with the application

The following tests will be performed:

1. Logon test - this will test the functionality of the 2 different types of logon ID's
    a. Login with DBA ID using DBA button on logon screen
    b. Login with SysAdmin ID using SysAdmin button on logon screen

c. Ensure users without accounts cannot log in

2. User Interface – This will test the displayed data and ensure it displays correctly and supplies the information the user is looking for.
    a. All data is read only for both ID's
    b. Able to scroll and navigate to the next page of information as both ID's
    c. Fields and systems are all displayed related to ID used to log on with
    d. Data is in a readable format
3. Functionality test - This will test the displayed features of data and functions within the application.
    a. Search function
    b. All fields are read only

Pass / Fail Conditions

It is expected that the Infrastructure Manager application must pass all tests in order for it to be a success. As stated before, most features have been undergoing testing as they are implemented, but a final test must be run of everything in order to ensure it is operating as expected.

Logon Test

Tester	Date	Item #	Expected	Actual	Pass/Fail	Bug
DBA Login	1/25/17	1.a	Login	Success	Pass	

SysAdmin Login	1/25/17	1.b	Login	Success	Pass	
Unauthorized User	1/25/17	1.c	Error Message	Success	Pass	

User Interface

Tester	Date	Item #	Expected	Actual	Pass/Fail	Bug
DBA Login	1/18/17	2.a	All data is read only	Success	Pass	
SysAdmin Login	1/18/17	2.a	All data is read only	Success	Pass	
DBA Login and SysAdmin Login	1/18/17	2.b	Scroll screen and navigate to next screen	Success	Pass	
DBA Login and SysAdmin Login	1/18/17	2.c	Information displayed pertains to logon ID	Success	Pass	
All Logins	1/18/17	2.d	Data is in readable format	Success	Pass	

Functionality Test

Tester	Date	Item #	Expected	Actual	Pass/Fail	Bug
All Logins	2/7/17	3.a	Search function for systems	Success	Pass	
All Logins	2/7/17	3.b	Data cannot be modified	Success	Pass	

Conclusion

This project has given us an opportunity to save a lot of time with administrative day to day tasks here at UC. This can save a couple of hours a day by not wasting time logging into several different applications or systems to find out simple information. This has made us excited to complete this project so we can use it here at UC. We have learned how to utilize a lot of the resources UC has already paid for which is a big plus for us, and the university. Also we have learned to utilize node.JS and some other utilities like WinSCP and Nagios. We are still looking to finalize some features and may add others later, but are really looking forward to making this available for others to use since the information provided can be very useful for troubleshooting and resource purchasing decisions.

References

"Useful Custom Command:: WinSCP" , 3 Mar. 2017

https://winscp.net/eng/docs/custom_commands#fxp

Poot, Jaap "AMIS Technology Blog" , 3 Apr. 2016

https://technology.amis.nl/2016/04/03/first-setup-connection-node-js-oracle-database/

Goode, Troy "CORS" *NPM*, 16 Jul, 2016

https://www.npmjs.com/package/cors

Code Samples

Table generation code used in angularJS

Name	Table Count	Row Count	MB Free	MB Used	MB Allocated	Date
{{db.DBNAME}}	{{db.TABLE_COUNT}}	{{db.ROW_COUNT}}	{{db.MB_FREE}}	{{db.MB_USED}}	{{db.MB_ALLOCATED}}	{{db.AS_OF_DATE}}

Routing for JSON server component to add data to tables

```
var express = require('express');
var cors = require('cors');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');

var routes = require('./routes/index');
var users = require('./routes/users');
var stats = require('./routes/stats');
var extstats = require('./routes/extstats');
var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

// uncomment after placing your favicon in /public
//app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
app.use(cors());

app.use('/', routes);
app.use('/users', users);
app.use('/stats', stats);
app.use('/extstats', extstats);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handlers

// development error handler
// will print stacktrace
if (app.get('env') === 'development') {
  app.use(function(err, req, res, next) {
    res.status(err.status || 500);
    res.render('error', {
      message: err.message,
      error: err
    });
  });
}

// production error handler
// no stacktraces leaked to user
app.use(function(err, req, res, next) {
```

```
  res.status(err.status || 500);
  res.render('error', {
    message: err.message,
    error: {}
  });
});


module.exports = app;
```

Web server routing for data to be displayed in the application

```
'use strict';

// Define statsApp
var statsApp = angular.module('statsApp',['datatables','ngRoute']);
statsApp.config(function ($routeProvider, $locationProvider) {
        // configure the routing rules here

      $routeProvider.when('/', {
            templateUrl: '/index.html',
            controller: 'HomeController'
        });


      $routeProvider.when('/dbHome', {
            templateUrl: '/dbHome.html',
            controller: 'StatsListController'
        });

        $routeProvider.when('/ext-stats/:dbName', {
            templateUrl: '/:dbName',
            controller: 'ExtStatsController'
        });

        // enable HTML5mode to disable hashbang urls
        //$locationProvider.html5Mode(true);
        $locationProvider.html5Mode({
          enabled: true,
          requireBase: false
         });
      });

// Define the controller in the module
//
//statsApp.controller('HomeController', function HomeController($scope, $http) {
//  .error(function(error, status, headers, config) {
//     console.log(status);
//     console.log("Error occured");
//  });
//});

statsApp.controller('StatsListController', function StatsListController($scope, $http) {
  $http.get('http://192.168.237.95:3000/stats')
  .success(function(data, status, headers, config) {
     $scope.databases = data;
  })
  .error(function(error, status, headers, config) {
     console.log(status);
     console.log("Error occured");
  });
});

statsApp.controller('ExtStatsController', function ExtStatsController($scope, $http,
$routeParams, $location) {
  var databaseName = $location.search().dbName;
  $scope.params = $routeParams;
```

```
  //var databaseName = $scope.params.dbName;
  console.log(databaseName);
  $http.get('http://192.168.237.95:3000/extstats/' + databaseName)
  .success(function(data, status, headers, config) {
     $scope.databases = data;
  })
  .error(function(error, status, headers, config) {
     console.log(status);
     console.log("Error occured");
  });
});
```