

TAPINSYSTEMS

Automating Cloud Services

Using Petri Net modeling techniques for quick
deployment and simplified maintenance

Peter Loh – Founder, Tap In Systems

November 12, 2009

Email: ploh@tapinsystems.com

Web: www.tapinsystems.com

Automating Cloud Services

The advent of cloud computing services has given IT organizations more options in how they can provide services to their customers. Cloud vendor services, such as Amazon Web Services, can offer benefits such as lower costs and shorter deployment times. Unlimited cloud compute resources can be used to add capacity to application processing workload “on-demand”. Off-site cloud storage offers “unlimited capacity” data backup options to on-site storage. This article describes Tap In System’s approach for effectively integrating these cloud services into enterprise IT processes and policies.

Systems Automation Issues

In most enterprises, IT services are not deployed independently. IT services are part of complex interaction of people, process and technology that have evolved over time. When evaluating the applicability of cloud services, one must ask how these services fit into the organizations current IT processes and policies. How can IT operations effectively manage these new services?

The method that cloud vendors promote for deploying their services is via programming language APIs. Their expectation is that end users, 3rd party vendors or consulting companies can use these APIs to integrate cloud services into usable applications. It’s all a simple matter of programming. However, historically, deploying systems automation (or its synonyms - run book automation, data center orchestration, IT operations automation) as a method for optimizing efficiency, improving availability, or reducing operator workload has been difficult. Systems automation has always been approached as a software application development project with the following traits.

- The project team consists of end users, operations, system administrators and application programmers. Certain systems management tools may make this process easier by standardizing on languages, protocols or APIs, but it still comes down to writing code.
- Though requirements may be derived by the needs of the IT operations organization, the overall system design is driven by programmers since they understand how to build the application. Though IT operations organization has the operational expertise in how the automation should work, they rarely have the skill set to develop the software. That is because writing software requires expertise in areas unrelated to practical IT operations – such as code language expertise, coding tools, use of repositories, coding best practices, build procedures.

- As with most software development projects, it takes a long time to implement new applications. Since the IT organization is constantly evolving, changes in requirements during the development process can cause the project to restart.
- Even after deployment and the application's original objectives are achieved, a process change will require a change in the automation application. This will probably require involvement of the original team members, no matter how trivial the change. Maintenance overhead is a major reason why the systems automation is difficult to achieve on an ongoing and consistent basis.
- When automation is implemented, an element of trust must be established between operations, the people responsible for providing service, and the automation application, the program that will react to service affecting events. Without this trust, even a robust automation application may not be deployed because of operations thinks that automation can not be reliable and the impact of failure is catastrophic.

Software vendors offer tools that claim to aid in automating systems management by offering products that the customer can use to avoid code. However, these applications are typically:

- *Brittle.* These vendors try to encapsulate automation scenarios within their own application code, then allow customers to modify their input parameters. The viability of these tools depends on how well the customer has defined their application. Any variation outside their input parameters may invalidate their application. Since individual company IT processes are closely aligned with automation processes, there is a likely chance that vendor automation will require customization, which leads to the issues listed above.
- *Require the entire vendor product suite.* Large software vendors usually have different products for server management, network management, application management – and additional products for subsets within each, for example Windows monitoring, Linux monitoring, web application monitoring, database monitoring, etc. Since systems automation requires coordination among various components, technologies, and application programming interfaces, vendors can more easily do this (and optimize sales!) if the customer is using all the vendor's products that perform individual tasks.
- *Expensive.* Because of the complexity involved in developing these applications, these tools can easily cost hundreds of thousands of dollars.

Systems Automation in the Cloud

Cloud services in particular are difficult to automate. Most applications that support cloud services are specific to one vendor's API, whereas applications which an enterprise might consider valuable require access and control across heterogeneous systems.

Examples of these applications include the following.

- *Automating recovery of cloud resources.* The value of cloud compute services, such as Amazon EC2, is that they can be started under program control. Virtualized systems enable resources to be assigned dynamically. Automating recovery requires interaction between the monitoring system, which should be aware of system alerts, and the cloud services APIs, to restart the resource. Tie ins to IT process may include creating problem tickets and notifying operations personnel when recovery occurs in order to track service levels.
- *Auto-scaling based on compute load.* This is a variation of the recovery scenario. In this case, the auto-scaling (scale up or scale down) action depends on the load on the application. The load calculation is explicitly or implicitly derived from the monitoring system metrics, which triggers the process to start a compute instance using the cloud API. The additional complexity comes from integrating interaction with other application components. For example, if adding web server capacity, the load balancer or application components may need to be reconfigured to define the additional web server resource. If IT processes dictate that configuration changes must be reflected in a configuration database, the auto-scaling process must interact with the configuration database.
- *Cloud storage backup.* Since cloud storage is theoretically “unlimited” and off premises, it seems to be a good candidate for backing up data center storage. The automation application must coordinate the storage capacity triggering alert from the monitoring system, the application control commands, and the cloud API action to initiate transfer to the cloud storage. Data storage applications that track critical application data would need to be notified.
- *Hybrid application.* Enterprise IT services may also be best served by hybrid applications – that is applications that have resources in the cloud and in their own data center. For example, cloud compute services, for scalable components, may be combined with data center components, which optimize high speed network connections. The interaction between cloud and non-cloud components can be governed by an automation application.
- *Cross vendor services.* One of the disadvantages of using a cloud service is that the vendor's service is a possible single point of failure. One way to mitigate that is to spread your resources across multiple cloud services. The automation application must be able to coordinate activity across these services, including:
 - Interfacing to multiple cloud service APIs.

- Recognizing vendor failures, versus network or component failures.
- Reconfiguring components to work with either redundant cloud service.
- Planning for and handling changes and failure scenarios in both sets of services.

Since cloud services are essentially extensions of data centers, developing cloud automation applications have the same issues as system automation applications, with the added complexity of dealing with multiple vendor interfaces.

Tap In Systems Approach

Tap In System has developed an approach toward systems automation with the goals of:

- Easing the systems automation development process.
- Easing the maintenance of systems automation applications.
- Providing ownership of automation to the IT operations organization.

The issues with developing automation have been present since systems management tools were used over 30 years ago. To address these issues, a change in philosophy is required – that is to change the ownership of automation to the IT operations organization. That is where the operational expertise exists, and therefore where ownership of automation, from an organizational perspective, should lie. However, the task of developing systems automation requires so much programming expertise, the development falls to the application development group.

In order to correctly align these responsibilities, the systems automation process should be divided into two parts – *modeling*, which is owned by operations, and *programming*, which is owned by development. Historically, the systems automation processes have been described in operational procedures or run book documentation. Modeling these procedures will require describing these processes in a more precise and scientific manner than English documentation. Tap In's approach is to define these processes as *Petri Nets* models.

Information and tutorials on Petri Nets can be found on the following links:

- Petri Net Worlds - <http://www.informatik.uni-hamburg.de/TGI/PetriNets/>
- Wikipedia - http://en.wikipedia.org/wiki/Petri_net
- Application of Petri Nets to Workflow Management - <http://is.tm.tue.nl/staff/wvdaalst/publications/p53.pdf>

Petri Nets are a proven method for describing how complex systems operate. Decades of research have been invested in examining the applicability of Petri Nets across a wide range of engineering disciplines. For IT processes, Petri Nets sufficiently describe how operational processes should work, yet do not require programming expertise. They require more rigorous definition than English-based documentation; however that

detailed knowledge is contained in the operations organization. Once the models are defined, they can be passed to application developers to code.

Maintenance changes also are controlled by operations by modifying the model. This methodology reduces involvement of application developers until specific changes are defined enough for developers to implement. The overall result is quicker and easier development process and maintenance. The methodology requires greater analytical expertise within operations - though not programming expertise – but better aligns organizational ownership.

Tap In's products enable this process in the following ways.

- Codifies the model into a file that can be created, saved and executed.
- Eases the model process by providing graphical tools for creating, debugging, and verifying models.
- Enables models to be linked to real time monitoring events – essentially transforming models from documentation to executable tasks.
- Allows operators to see model execution and results.
- Provides a programming framework to link models to code development and execution.
- Provides an execution framework where models can be scheduled and run in a production environment.

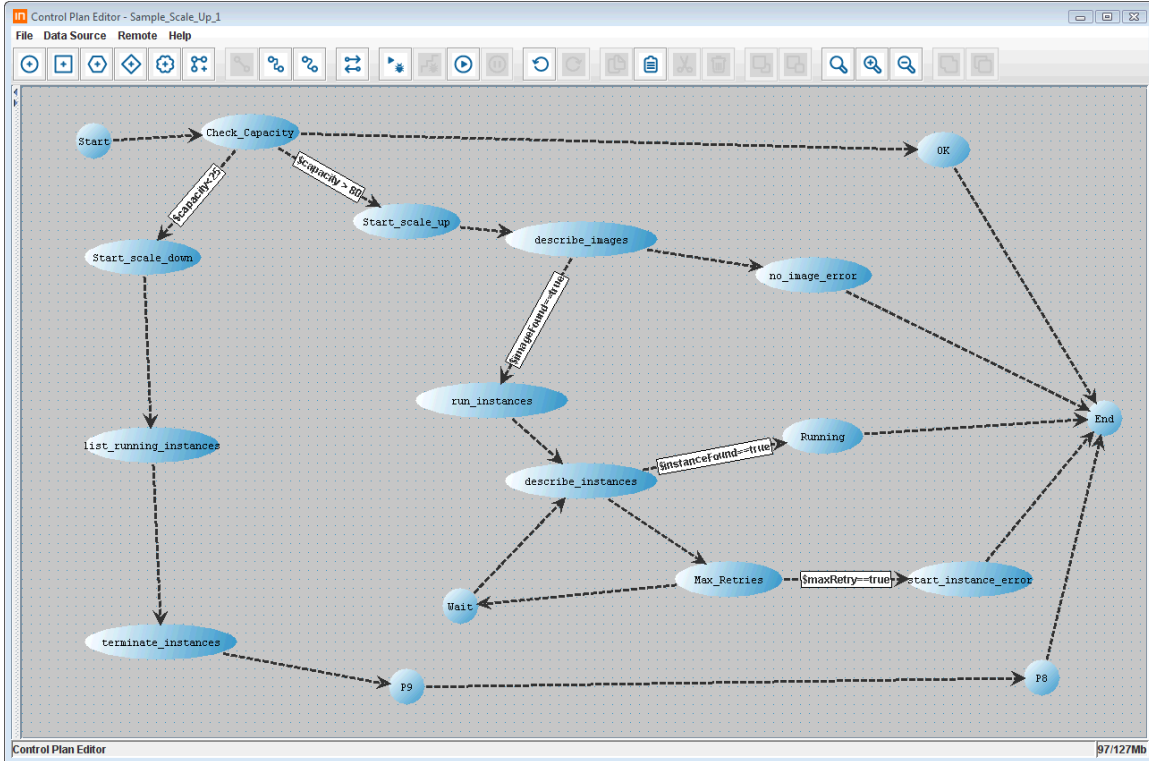
These features are implemented in Tap In's Control Plan Editor product. Its graphical interface allows the operator to easily create Tap In's version of a Petri Net model, called a Control Plan. Control Plans extend the Petri Net model by allowing work to be executed at each place in the model. This work is an automation task, such as calling a cloud API, reading database values, or executing a recovery action. Place tasks are prepackaged and provided by Tap In Systems, or can be customized by the user. The triggers in the net can be initiated by the results of the work, including examining events from Tap In's Cloud Management Service (CMS). CMS monitors events from all components in the infrastructure.

When a Tap In Control Plan executes, the net model changes color to reflect the current state of the net and which place is currently executing. A console output allows work tasks to print detailed status information.

Cloud Auto-Scaling Example

Let's look at an example of how a systems automation task can be implemented using this approach. One common automation request for cloud applications is to be able to automate the scaling of servers according to load. If load (the user can define and calculate load) exceeds a threshold, add capacity. If is below a threshold, reduce capacity.

Using Tap In's Control Plan Editor, the Petri Net below is created.

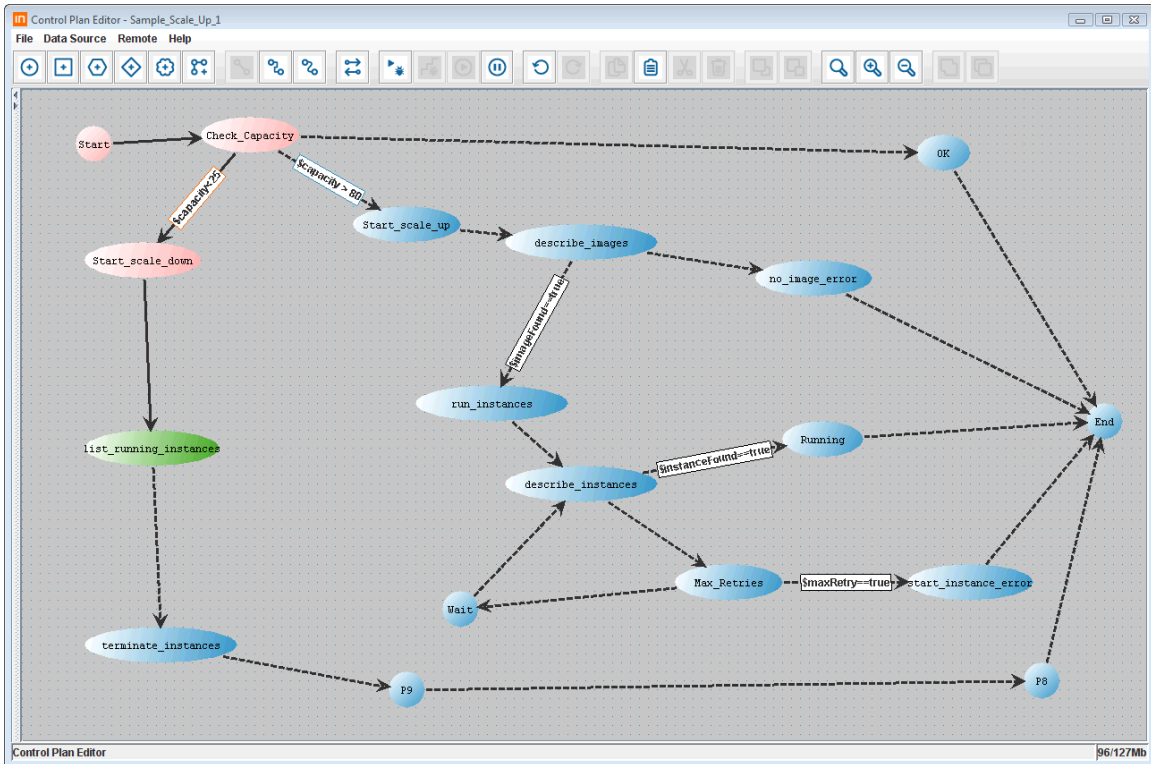


The Control Plan Editor's user interface can be used to create places and connect transitions by drag and drop actions. The Control Plan shows the workflow for auto scaling. The places shown include the following work tasks:

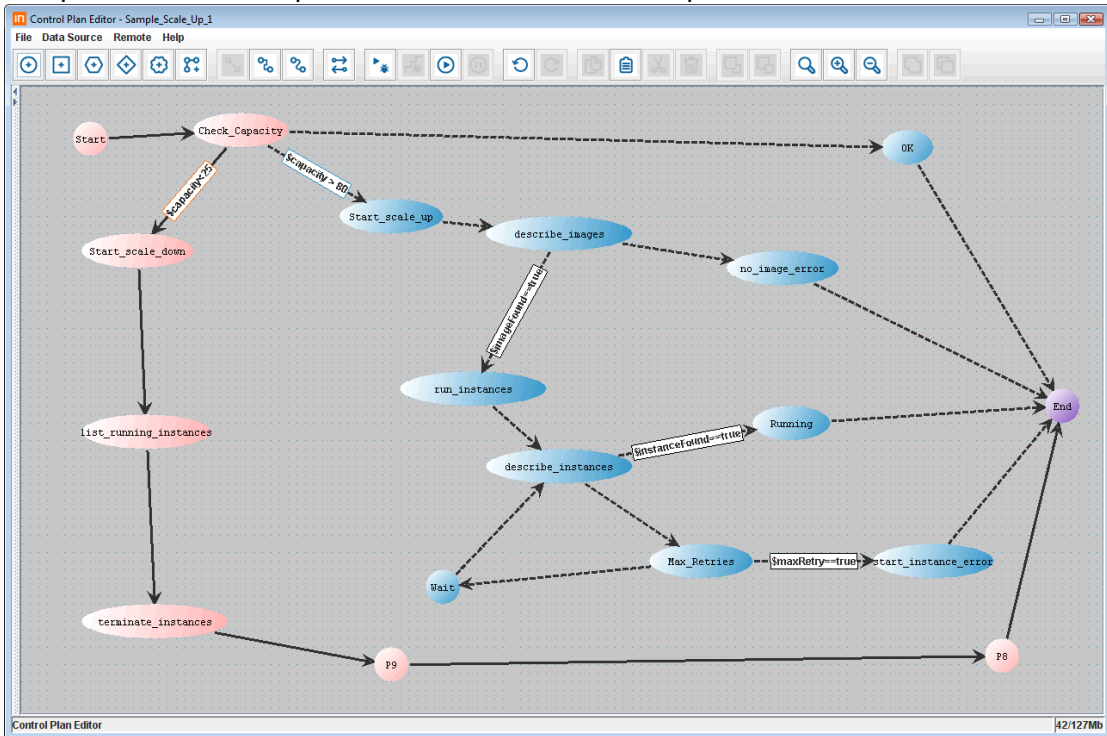
- Check_Capacity – read events from the Tap In monitoring system that indicate the load thresholds are reached.
- List, run and terminate instances. These initiate APIs calls to the cloud system to list the running compute instances, start a compute instance or terminate a compute instance. Note that because these API calls are encapsulated with places in this plan, they can easily be replaced with calls to an alternate cloud vendor if needed. This demonstrates separation between an internal IT process and supporting vendor cloud services.
- Error reports. These places generate error messages to monitoring tools or notification methods. This ensures communication between the automation and operations personnel.

If we execute this plan, Control Plan will show what tasks were executed by color coding the plan. As shown below, the load was below the capacity threshold so the scale down

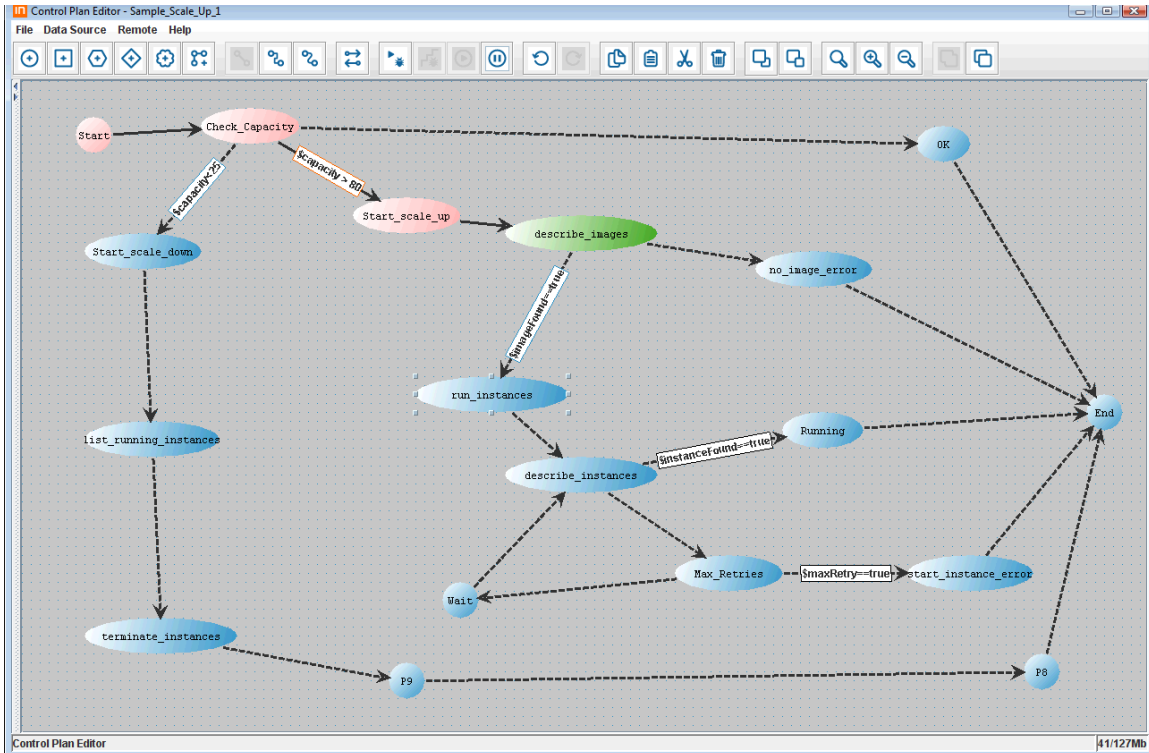
tasks were taken. This plan is executing the “terminate_instances” task as part of the scale down process.



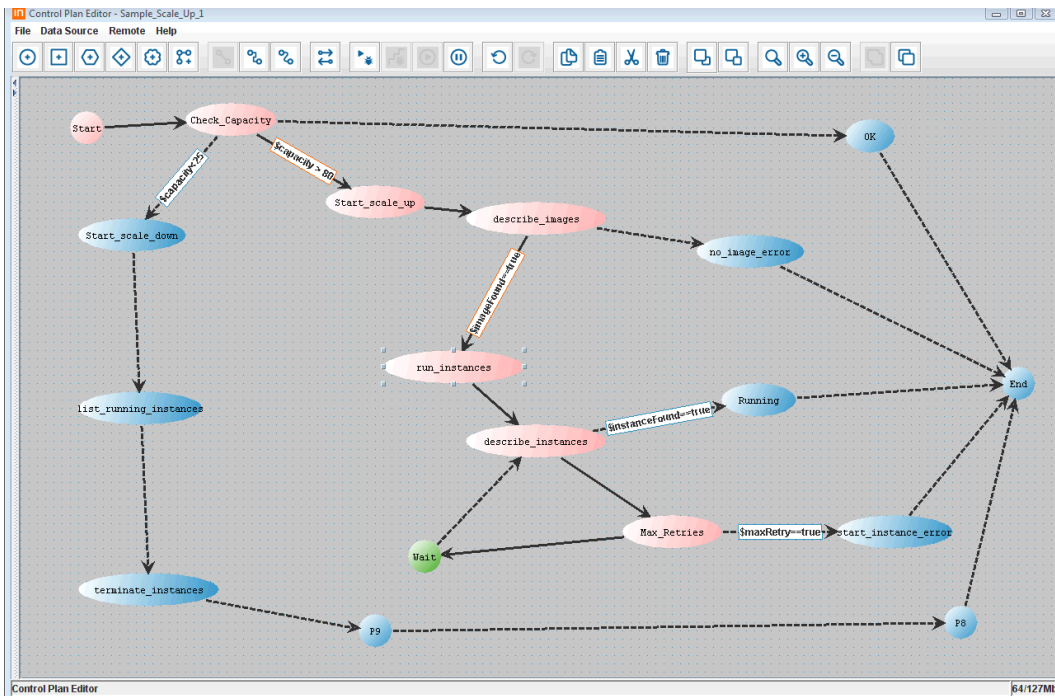
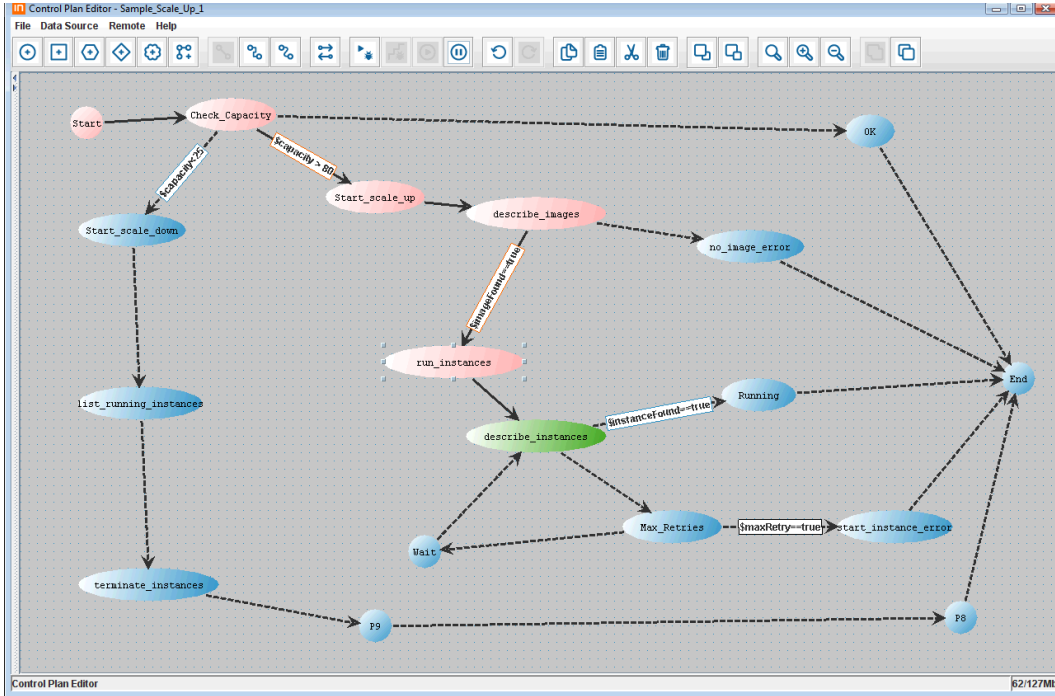
The plan has now completed since it is at the “end” place.



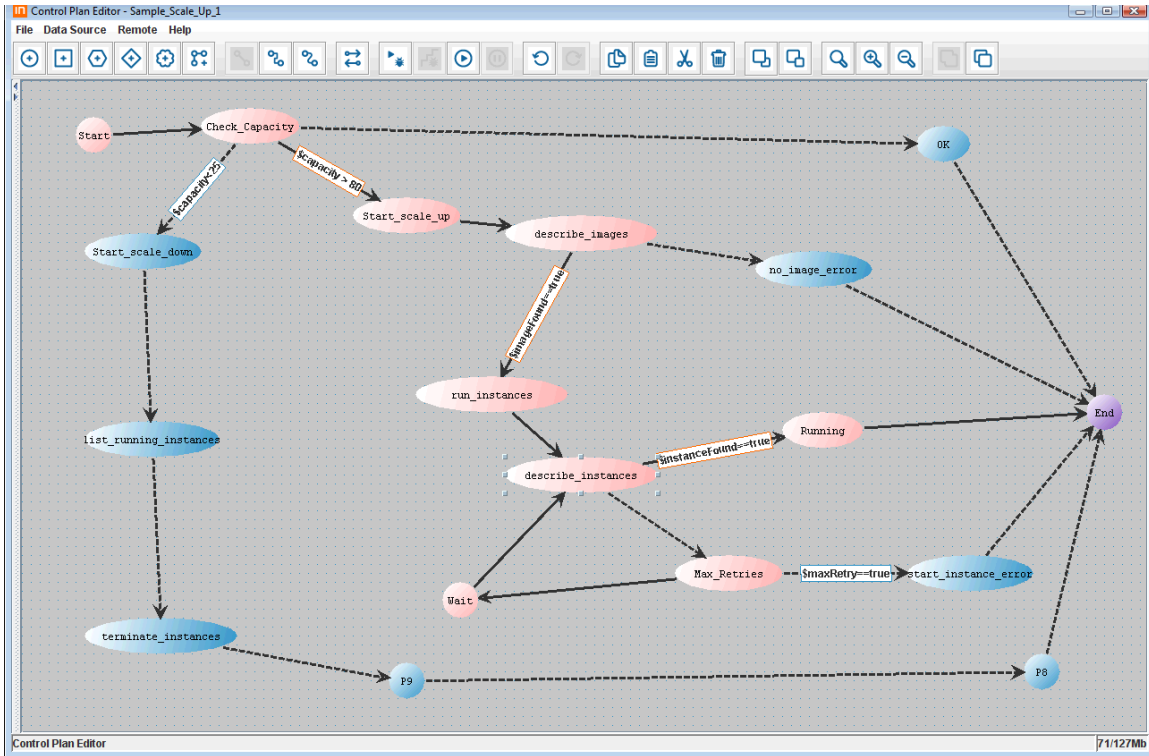
An alternate possibility when the plan executes is if the load is above capacity, indicating the scale up process should be taken. In this case, the plan will first list all the images (“describe_images”) in order to see if the requested image to start actually exists. If it does, an instance of this image will be started. The plan will check to make sure the instance goes to a running state by checking periodically. The diagram below shows the plan starting the scale up process by checking for existing images.



In this case the image was found and the instance was started. The plan is now waiting for the instance to go to “running” state.

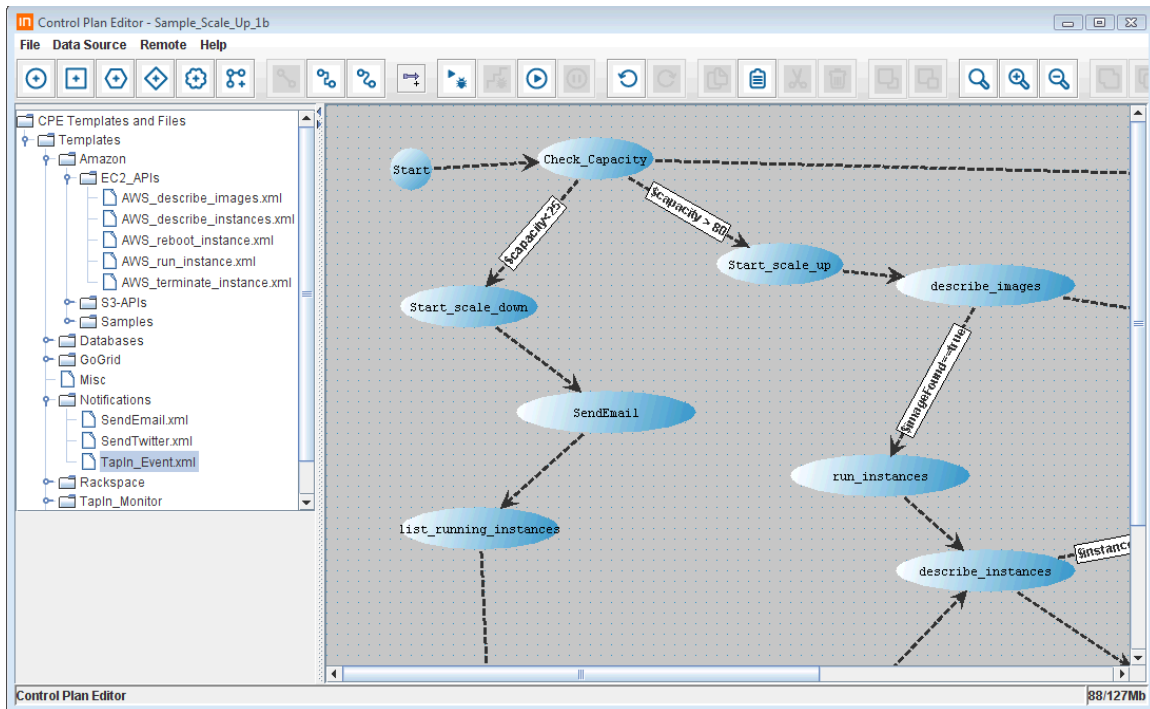


The instance went into running state then generating an instance successfully running notification before ending.



The advantages of using Control Plan over other automation application implementations are clear.

1. The models can be easily created by using the Control Plan graphical editor. Since the model shows operational processes rather than programming constructs, it is feasible for skilled operations personnel to own the plans.
2. Each place task can be a prepackaged module or a piece of code (Ruby or Java). Application developers can concentrate on their areas of expertise – the execution of APIs and interaction between program components. This maximizes the use of resources.
3. Operators can better trust these actions since they create them and test them. They can also see plans as they execute, as shown above. Operators can also ensure automation tasks communicate status properly by inserting the proper notification tasks in their plan.
4. Control Plans are much easier to maintain. Predefined tasks can simply be added by modifying the plan. The following screen show shows the templates that are available to a plan.



To add a place tasks to the model, the user can just drag the template to the canvas. For example, to add an email notification when an error occurs, just drag the email Notification template to the canvas, and place it within the error section of the net.

This ability for operations to maintain automation plans is a key advantage. Even if additional programming is required to add a new place task, the entire automation plan does not have to be re-written. The developer can create the new place task and allow the operator to place it within the model.

The development approach using Control Plan models can be very flexible. The operator can use a top-down modeling approach by defining the application states he is interested in, then deriving the actions and logic required to determine those states. Once the basic high level model is developed, more complex logic can be used by replacing individual places with other Control Plans. A plan that can be used as a place in another plan is called a supernode.

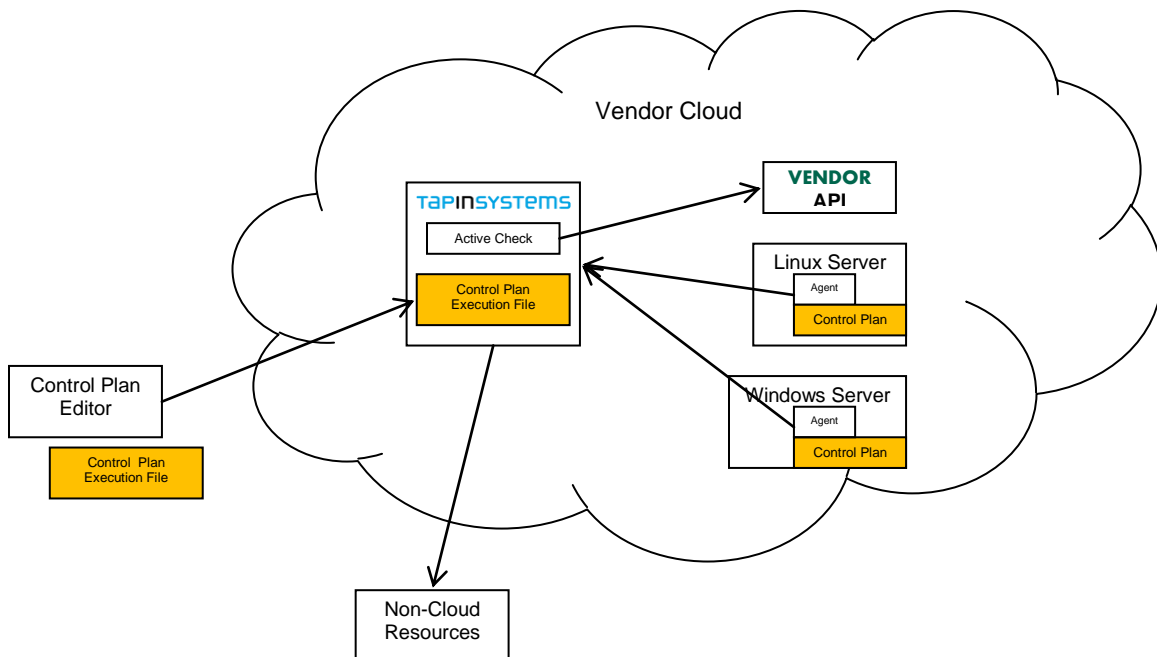
In the above example, we have a single place that calculates whether the capacity is exceeded by examining the current load of the system. But suppose this decision point is more complicated than looking at a simple variable. For example, the logic might involve an algorithm looking at server load, current number of users, time of day, day of week, current number of web servers in production, database connections used, etc. We can create a Control Plan that encapsulates all these variables, whose output is whether capacity is over threshold, under threshold or OK. This plan can be defined as a

supernode for the check_capacity place in the higher level diagram. This capability allows layering of plans to create complex models. Yet, the operator can hide the complexity from high level plans if desired

Production Deployment

After they are created, Control Plans can be used deployed in a production environment in a number of ways. In the simplest case, since the Control Plan Editor executes the plan, an operator can choose to execute plans only under operator control. If operator run book procedures are defined as plans, the operator can use Control Plan as a way to execute manual problem diagnostic/recovery procedures automatically and see the results graphically. Control Plan features, such as dialog boxes, can enhance this functionality.

However, Control Plans can also run independent of the graphical editor, as unattended programs, with the Control Plan Server. This scenario is shown in the diagram below.



The Control Plan can execute on any distributed node without its graphical user interface. Of course if designed to work in this manner, the plan should not contain any user interaction functions like dialog boxes, and should notify operators of status via other mechanisms, like events to Tap In’s monitoring system. This configuration allows

automation to be executed where appropriate, which is typically “close to the source” of the problem. Since multiple plans can be scheduled to run on any system, a plan for each automation scenario can be configured to run.

Summary

This paper outlines a new approach for automating systems management and cloud applications. By employing Petri Net modeling techniques, IT operations personnel can assume the responsibility of automation applications, which results in quicker development and easier maintenance. Involvement of application developers, which are scarce resources, is minimized and more efficiently applied. Codifying operational procedures into Petri Net models allows complex processes to be developed in a stepwise and rigorous manner.

Tap In’s Control Plan Editor can be used to easily define and deploy these automation models. Its graphical interface eases creation and modification of these models. Real time execution results can also be viewed. These tools enable user to take advantage of cloud services by integrating enterprise IT processes with cloud service functions.

For additional information about any of the topics discussed in this paper, contact Tap In Systems at info@tapinsystems.com.