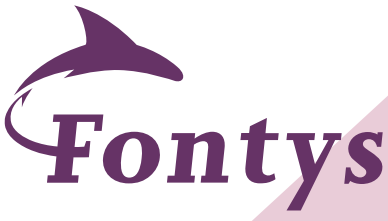
Analysis and Considerations for Using MATLAB Production Server

Tomasz Olejarczuk



23-04-2024

Contents

Analysis and Considerations for Using MATLAB Production Server

1. Summary

MATLAB Production Server addresses the limitations of server-based MATLAB deployments that rely on file-exchange mechanisms. It enables seamless integration of MATLAB algorithms into production environments, offering significant advantages over MATLAB Standalone Archives in terms of scalability, performance and architectural flexibility.

2. Introduction

MATLAB Standalone Archives offer a way to distribute MATLAB code without requiring a full MATLAB installation on the end-user's machine. However, they present limitations when deployed in server environments, especially where scalability, performance and integration with other systems are crucial requirements.

MATLAB Production Server addresses some of these challenges by providing a more streamlined way to deploy and execute MATLAB algorithms on servers. Here's an overview of what it does and the advantages it offers:

- **Packaging and Deployment:** MATLAB Production Server helps you package MATLAB algorithms and necessary dependencies into deployable archives. These can then be more easily deployed on the server, making the algorithms ready for execution.
- **Integration and Communication:** MATLAB Production Server exposes an endpoint for your deployed algorithms. This allows external applications (including those written in languages like Java, .NET, Python, or C/C++) to call these algorithms and utilize the computational power of MATLAB.
- **Scalability and Performance:** While the specifics might depend on your setup, MATLAB Production Server can potentially handle a higher volume of requests than a pure file-exchange system and its deployment model provides some efficiency gains.

2.1. Matlab Production Server vs Matlab Standalone Archives

MATLAB Production Server does provide specific benefits in the context of server deployments:

- **Integration Potential:** This remains the most significant advantage. MATLAB Production Server provides an endpoint for your MATLAB algorithms. This allows for integration with other applica-

tions or processes through direct function calls and offers more flexibility than what you can achieve with standalone executables.

Important Considerations

- **Licensing:** While both MATLAB Production Server and Standalone Archives offer ways to execute code without a full MATLAB license on some machines, there still may be licensing requirements for the machine where the MATLAB Production Server itself is installed or if deploying in a cloud environment.
- **Scalability:** True scalability with MATLAB Production Server requires manual deployment of multiple server instances and likely the use of a load balancer. It does not inherently provide automatic scaling or sophisticated resource management on a single server instance.
- **Deployment Workflow:** If utilizing Docker containerization, the streamlined packaging workflow offered by MATLAB Production Server may not provide a significant advantage, as both Standalone Archives and the MATLAB Production Server itself can be packaged in such a way.

3. Methodology

3.1. Technical Overview

The process of creating the MATLAB Production Server archive is automated using a custom MATLAB build script. The script uses these key functions:

```
function build()

res = compiler.build.productionServerArchive('teamiumalgorithmjson.m',
    'ArchiveName', 'teamium', "Verbose", "On", 'SupportPackages','
    autodetect')
compiler.package.microserviceDockerImage(res);

end
```

3.1.1. Build script code breakdown

1. **Building the standalone application**

   `compiler.build.standaloneApplication` - this command builds the standalone application

- **Parameters:**

  - `'teamiumalgorithm.m'` - entrypoint file
  - `'SupportPackages'`, `'autodetect'` - Matlab takes those parameter as a name of the parameter and it's value, so it autodetects which `SupportPackages` are needed automatically

```
res = compiler.build.standaloneApplication('teamiumalgorithm.m', '
    SupportPackages', 'autodetect');
```

2. **Building the Docker image**

`compiler.package.microserviceDockerImage` - This function encapsulates the production server archive and its runtime environment into a self-contained Docker image

- **Parameters:**
  - `res` - The previously created MATLAB Production Server archive is passed as input, ensuring that it becomes the core of the image.

```
compiler.package.microserviceDockerImage(res);
```

4. Results

The adoption of a MATLAB Production Server archive has resulted in the following improvements:

- **Improved Integration:** The provided endpoint for the MATLAB algorithm makes integration with external applications or processes more structured compared to a pure file-exchange mechanism. External systems can interact with the algorithm via function calls that contain input data in JSON format.
- **Simplified Architecture:** The transition to a Production Server model reduces some of the architectural complexity associated with the previous file-exchange based approach. Direct execution of the core algorithm eliminates the need to manage files as an intermediary step.

Important Considerations

- **Scalability Limitations:** MATLAB Production Server, while an improvement over file-exchange, does not inherently solve the scalability challenges of server-side MATLAB algorithm execution.
- **Image Transfer:** The current workflow still relies on a file-exchange mechanism for image data transfer. This could become a potential bottleneck as the system handles larger images or increased request volume.

5. Conclusion

The use of a MATLAB Production Server has brought valuable improvements to the deployment and execution of your MATLAB algorithm within your server environment. The streamlined packaging process, simplified deployment using Docker, and the ability to interact with the algorithm via its endpoint provide a more cohesive and manageable foundation for integration within your production systems. While the current workflow still relies on file-exchange for image data transfer, the MATLAB Production Server offers advantages over a purely file-based approach. As the project grows, continuous evaluation of its architectural needs, especially relating to scaling image transfer and computational requirements, will be essential to ensure the chosen MATLAB deployment strategy remains the best fit.

6. Next Steps

To maximize the capabilities of your current MATLAB Production Server deployment and pave the way for future scalability, consider these next steps:

- **API Wrapper Development:** Develop a custom API wrapper that acts as the central point for interaction with your system. This API will primarily function as an image retrieval and delivery service.
  - **Image Retrieval:** The API can accept requests through its defined endpoints to retrieve the pre-saved PNG image generated by your MATLAB Production Server.
    * **Error Handling:** Implement robust error handling to ensure successful retrieval of the image file and handle potential issues like file corruption or missing files.
    * **Security (Optional):** If the image data is sensitive, explore authentication and authorization mechanisms for the API to ensure data security.
  - **API Response:** Define a clear response format for the API to deliver the retrieved PNG image data to the requester. This might involve simply delivering the raw image data or potentially embedding it within a JSON response structure.
- **Kubernetes Migration:** As your project's computational demands increase, consider migrating your deployment to a Kubernetes cluster. Kubernetes offers robust container orchestration features, including:
  - **Scalability on Demand:** Easily deploy and manage multiple instances of your MATLAB Production Server, scaling up or down dynamically based on workload volume.

  - **Load Balancing:** Integrate a load balancer to distribute incoming requests across your MATLAB Production Server instances, preventing any single instance from becoming a bottleneck.
  - **Resource Management:** Kubernetes intelligently allocates and manages compute resources (CPU, memory) within your cluster, ensuring your MATLAB Production Server pods have the resources they need to perform optimally.

- **Image Transfer Optimization (Optional):** While your current workflow utilizes file exchange for image transfer, explore alternative methods like container registries if this process becomes a bottleneck for frequent deployments. Container registries offer a more centralized and scalable approach for managing Docker images.

Prioritization: Developing an API wrapper with secure image retrieval and a well-defined response format can be a near-term solution for improved integration and efficiency. A Kubernetes migration might be a long-term goal for significant scaling needs.

Transitioning to Kubernetes would bring a level of scalability and automation that is essential as your project grows in scope and demand.